Perfect-Information Games with Cycles

Daniel Andersson

PhD Dissertation

Department of Computer Science
Aarhus University
Denmark

Perfect-Information Games with Cycles

A Dissertation
Presented to the Faculty of Science
of Aarhus University
in Partial Fulfilment of the Requirements for the
PhD Degree

by
Daniel Andersson
July 31, 2009

Abstract

This dissertation presents algorithmic and complexity-theoretic results on games, puzzles and optimization. The major theme is perfect-information games played on finite directed graphs.

- We show that Washburn's deterministic graphical games—a natural generalization of Zermelo's chess-like games to arbitrary zero-sum payoffs—can be solved in almost-linear time using a comparison-based algorithm.
- We establish polynomial-time equivalence of solving several well-known classes of zero-sum games with moves of chance: stochastic games with terminal payoffs, mean payoffs, discounted payoffs, or parity conditions.
- Motivated by their connection to positional Nash equilibria, we investigate the concept of strategy improvement cycles in deterministic multi-player games with terminal payoffs. We obtain partial results and outline the limits of this technique.

A minor theme is the complexity of puzzles, following a long tradition of hardness results for well-known puzzle types. Hiroimono is a 600-year-old tour puzzle wherein stones are to be collected from a grid. We show that deciding the solvability of such a puzzle is NP-complete. A similar result is also shown for Hashiwokakero, wherein islands are to be connected by bridges.

Finally, we consider shortest path network interdiction—destroying parts of a network to maximize the length of the shortest path. Khachiyan et al. considered a model with vertex-local budgets of edge removals. We generalize their model and results by adding the capability of partial interdiction of edges. We also establish a connection to deterministic graphical games.

Acknowledgements

This dissertation concludes three years of study and research. I am deeply grateful to my advisor Peter Bro Miltersen for his constant support and encouragement throughout this time. His enthusiasm has been inspiring, and his frank and respectful guidance has been immensely helpful.

I thank my fellow PhD students and post-docs for making our department such an enjoyable workplace. Troels deserves special mention for putting up with me as his office mate for so long.

For their generous financial support I thank the taxpayers of Denmark. Special thanks also go to Google Denmark for providing a much-needed change of environment and for being patient while I finished this dissertation.

Last, but most, I thank Rocio, without whom I would be nowhere.

Daniel Andersson,
Aarhus, July 31, 2009.

Contents

Part I

Overview

Chapter 1

Introduction

1.1 Games

Game theory is the study of interactions that contain elements of competition or conflict. Informally, a *game* is any situation wherein two or more decision-makers together determine one outcome, but disagree on which outcome they prefer. As such situations are abundant, this is clearly a very general and potentially applicable concept. Game theory originated in the field of economics [58] but has been applied to a wide variety of subjects, ranging from evolutionary biology [53] to international relations [44].

In more detail, a game generally consists of the following basic ingredients.

1. A set of *players* I.
2. For each player $i \in I$, a set of *strategies* S_i from which player i must select one strategy.
3. A map from $\prod_{i \in I} S_i$ to a set of *outcomes* A. An element of $\prod_{i \in I} S_i$ is called a *strategy profile*.
4. For each player, a real-valued *payoff function* defined on A. Each player seeks to maximize his payoff.

This manner of specifying a game is called the *normal form*. Figure 1.1 presents a concrete example of a game in normal form.

The special case of a game with two players that have totally opposing interests, i.e., the gain of one equals the loss of the other, is called a *zero-sum* game (as the payoffs sum to zero in each outcome). Many of the games considered herein will have this property. When describing such a game, it is sufficient to specify the payoff for one of the players, typically Player 1, whom we then also refer to as "Max" (with Player 2 being "Min"). Even more of a special case are win/lose/draw games; to model these, we can simply define the payoff to a player to be 1, -1 or 0 in case he wins, loses, or there is a draw, respectively.

When a game has been defined, we can start to ask questions about it. These can be of a descriptive nature: Which strategies will the players select? (What will the outcome be?) Alternatively, we can adopt the viewpoint of a single

		Player 2			
		a	b	c	d
Player 1	A	(2, 3)	(4, 2)	(5, –3)	(–2, 3)
	B	(1, 4)	(–1, 3)	(–3, 2)	(0, 4)
	C	(3, –2)	(0, 2)	(0, 3)	(–1, 1)

Figure 1.1: A two-player game in normal form. Player 1 selects a row, and Player 2 simultaneously selects a column. The first and second components of the corresponding matrix entry is the resulting payoff to Player 1 and 2, respectively.

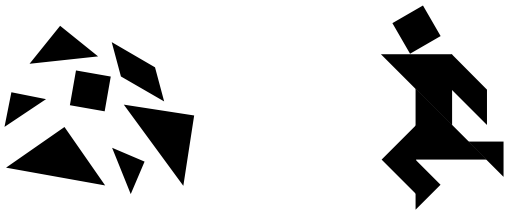


Figure 1.2: A traditional Tangram puzzle: Rearrange the seven pieces on the left to form the figure on the right.

player: If I were Player 1, which strategy should I select? These two viewpoints are intimately related, and some thought is required to make questions such as these mathematically well-defined.

1.2 Games vs. Puzzles and Optimization

Games are commonly associated (or confused) with *puzzles*. In our terminology, games are interactive and competitive, while puzzles are solitary. Informally, in a puzzle, your task is to search for some kind of *solution* that satisfies a given set of constraints. You will know when you have found it, and you can show it to other people to convince them that you solved the puzzle. For them, checking whether your proposed solution is correct is a much easier task than solving the puzzle from scratch. An example of a puzzle in this sense is a Tangram; see Figure 1.2.

With the definition of a game given above, a "one-player game" simply becomes another way to describe the well-known concept of an optimization problem. A classic example is the shortest path problem: Given a graph and two of its vertices s and t, find the shortest path from s to t. The strategies of the corresponding one-player game are the s–t paths, and the utility function simply gives the negated length of each path (translating minimization into maximization).

Both puzzles and optimization problems immediately lend themselves to fairly well-defined computational formulations: Given a puzzle, find a solution. Given a graph, find the shortest path. For games however, this is not so im-

mediate. We could try to mimic optimization problems: Given a game, find an "optimal" strategy for Player 1. Of course, the term "optimal" is *a priori* meaningless, since the outcome also depends on the other players, who in turn may reason about Player 1, and so on. We will however encounter some special cases where this term can be reasonably defined. A more puzzle-like approach would be to define some criteria for what outcomes are "likely", i.e., some constraints that a "reasonable" strategy profile should satisfy. We will consider both of these approaches.

Best Guarantees: Maximin Strategies. Let us take the viewpoint of Player 1 in the game in Figure 1.1. What would be a good strategy? If we select A, then we can in the best case get payoff 5, but in the worst case, if Player 2 selects d, we get -2. If we only care about the worst case, C is our best choice; it guarantees a payoff of at least -1. This is called a *maximin* strategy—it maximizes (among our strategies) our minimum (among Player 2's strategies) payoff. Although this approach may seem overly pessimistic, the guarantee we obtain is very strong, as it holds against any opponent, even an irrational one.

Stable Situations: Nash Equilibria. In a game, an *equilibrium* is a strategy profile which is in some sense "stable". Consider again the game in Figure 1.1 and suppose Player 1 chooses B and Player 2 chooses d. This strategy profile, (B, d), has the following stability property: No player can obtain a better payoff by changing his strategy (assuming all other players stay put). Such a profile is called a *Nash equilibrium*. In general, a game can have any number of equilibria, with differing payoffs.

In the case of zero-sum games, there is a connection between Nash equilibria and maximin strategies: If a zero-sum game has a Nash equilibrium (X, y), then X must also be a maximin strategy for Player 1 and y likewise for Player 2. Thus, if we denote by v the payoff to Player 1 resulting from (X, y), then Player 1 can guarantee to get at least v (by choosing X), and Player 2 can guarantee that Player 1 gets *at most* v (by choosing y). This unique number v is called the *value* of the game. If a game has a value, maximin strategies are sometimes referred to as *optimal* strategies.

Mixed Strategies. It is common to augment a game by considering *mixed strategies*—essentially allowing players to randomize their choice of strategy. Formally, a new game is defined, where the strategies of each player are all probability distributions on his strategies in the original game. Payoffs are defined as expectations of the payoff distributions induced in the original game. In this context, the term *pure strategy* is used to describe both the strategies in the original game and the distributions that put all probability mass in one point. Whenever we include mixed strategies, this will be explicitly mentioned.

Von Neumann [57] proved that zero-sum games with finite strategy sets that are augmented in this way always have a value. Nash [43] later generalized this to the existence of equilibria in n-player games. Although these are landmark

results in the theory of games, they will not be applicable to most of the games we will consider, as the strategy sets will be infinite.

1.3 Algorithms and Complexity

This dissertation treats many computational problems related to games, puzzles, and optimization. We seek to construct efficient algorithms for solving these problems or, failing that, establish some hardness results.

The algorithmic viewpoint we adopt is that of an "offline" problem—the entire input (e.g., a game) is given at once to the algorithm, which then, after some computation, produces the entire output (e.g., a maximin strategy). Our main criterion for evaluating an algorithm is its asymptotic *worst-case* running time as a function of input size. We use O, Ω, and Θ to denote upper, lower, and tight asymptotic bounds, respectively.

Unless otherwise noted, our algorithms run on a standard random access machine, with inputs somehow encoded as binary strings. In this case, payoffs, ideally arbitrary real numbers, are assumed to be rational to allow for straightforward encoding and computation. Sometimes we consider *comparison-based* algorithms, whose only access to the real numbers in their input goes through an "oracle" that answers queries about which of two numbers is greater. In this model, we may also study the number of such comparisons as a separate measure of efficiency and consider trade-offs between time and comparisons (see [12, 30] for other work in this spirit).

To investigate the relative hardness of problems we use reductions—if A can be efficiently reduced to B, then A cannot be much harder than B. Most of the time, we will use Turing reductions (also known as Cook reductions), where A is solved using an oracle that solves B. In our treatment of puzzles, we will use the stricter notion of a many-one (or Karp) reduction, where an instance of A is transformed into an instance of B having the same answer.

Chapter 2

Games on Graphs

2.1 Games with Several Steps

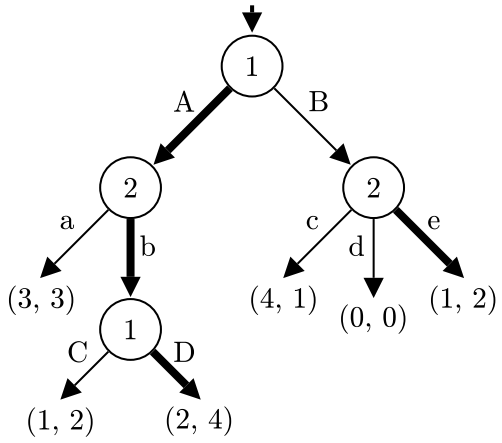
The normal form for games is very general, and most games, if not all, can be put into that form. However, for many games the normal form is not the most natural or practical. It is sometimes useful to consider models with more structure, where strategies are not monolithic and abstract. In particular, many games are played over time, with several *actions*, or *moves*, chosen in sequence, perhaps alternatingly between players. During the course of play, the set of available moves may change. A simple real-life example is Tic-Tac-Toe.

Trees. To model games with several steps, we can use the *extensive form*, also known as a *game tree*. Figure 2.1 shows an example. The nodes of the tree correspond to possible states of the world, called *positions*. Starting from the root, the players build a path through the tree, a *play*, down to a terminal position (leaf) which determines the resulting payoffs. Here, we will restrict our attention to the case of *perfect-information* games—when a player is to move, he knows the current position as well as the full history of all previous moves.

As shown in Figure 2.1, an extensive form game can be put into normal form simply by defining the strategy set of a player to be the Cartesian product of all sets of moves he may have to choose from. However, the size of the corresponding normal form representation grows exponentially in the size of the game tree.

It is easy to construct a Nash equilibrium in a game tree using the method of *backward induction* [58]: For each position in a bottom-up fashion, fix a move that maximizes the current player's payoff, assuming future play will follow the previously fixed moves. Figure 2.1 shows a Nash equilibrium constructed this way.

Acyclic Graphs. A game where the same state can be reached in more than one way (like Tic-Tac-Toe) is more conveniently represented using a directed acyclic graph, a *game graph*, rather than a tree. Figure 2.2 shows an example. In this model, a position only specifies the possible *future* of play, but does not encode the full history. However, the players still remember the full history, and



		Player 2					
		ac	ad	ae	bc	bd	**be**
Player 1	AC	(3, 3)	(3, 3)	(3, 3)	(1, 2)	(1, 2)	(1, 2)
	AD	(3, 3)	(3, 3)	(3, 3)	(2, 4)	(2, 4)	(2, 4)
	BC	(4, 1)	(0, 0)	(1, 2)	(4, 1)	(0, 0)	(1, 2)
	BD	(4, 1)	(0, 0)	(1, 2)	(4, 1)	(0, 0)	(1, 2)

Figure 2.1: A two-player game played on a tree (*top*). Each node represents a non-terminal position of the game, wherein the current player (indicated inside the node) selects one of its outgoing arcs. The topmost node is the starting position. Terminal positions are tuples specifying payoffs. In the normal form (*bottom*) representation of this game, the strategy sets are Cartesian products of sets of moves. A Nash equilibrium is indicated in bold.

may base their strategy on it. We can "unfold" the graph into an equivalent game tree, at the cost of a potentially exponential increase in size. We can however perform backward induction without such unfolding; we simply have to topologically order the positions so that we can process each position after all its successors. The Nash equilibrium thus constructed consists of strategies with a special property: The move made at a position is independent of how that position was reached. Such memory-less, or *positional*, strategies are particularly easy to represent (a fixed selection of one move per position) and analyze, and they are therefore of considerable interest. A Nash equilibrium consisting of only such strategies will itself also be called positional; Figure 2.2 shows an example.

2.2 Cycles and Infinite Play

Looking at the directed acyclic graph representation, one may find it natural to also consider arbitrary finite game graphs with cycles. Although this step

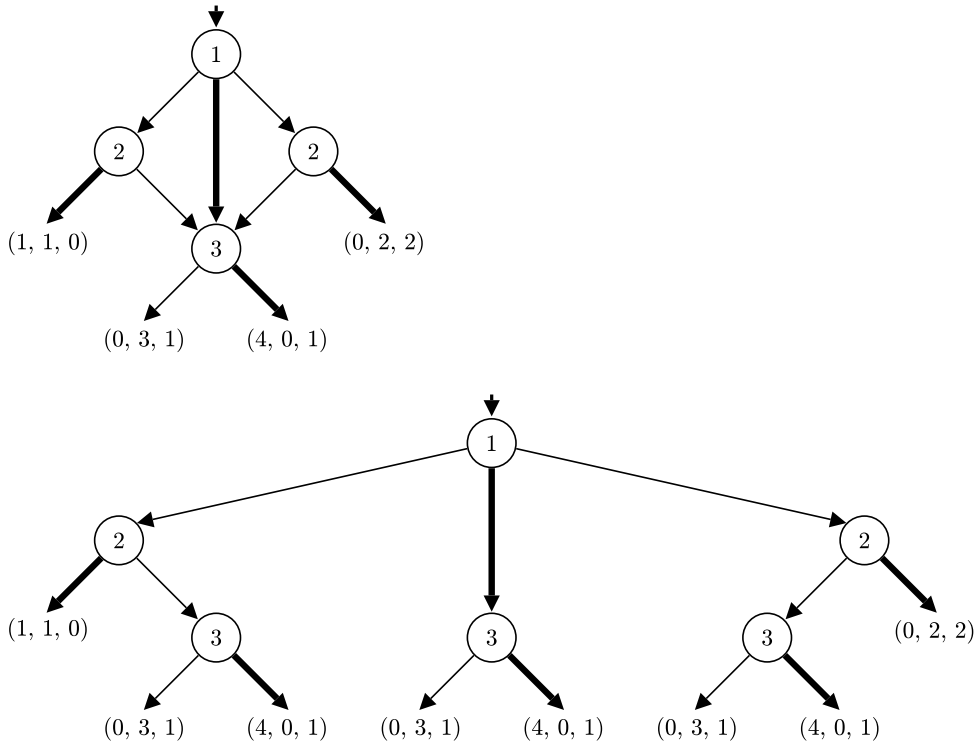


Figure 2.2: A three-player game played on a directed acyclic graph (*top*). It can be "unfolded" into an equivalent game tree (*bottom*). A positional Nash equilibrium (and its counterpart in the tree) is indicated in bold.

might superficially seem small, it has fundamental ramifications. A cycle in the game graph corresponds to the possibility of visiting some positions more than once, and in general, an unbounded or even infinite number of times. Thus, the sets of strategies become infinite (with the positional ones as a finite subset). The play may never end, and we must decide how to interpret this eventuality.

The study of games with cycles dates back to Zermelo's classic 1913 treatment of chess [61]. In his model, any infinite play is considered a draw. In modern terms, Zermelo set out to prove that chess has a value (i.e., either White can guarantee to win, or Black can guarantee to win, or both players can guarantee not to lose). Paper 1 provides a more detailed historical survey of the "chess-like" games captured by this model. In contrast, in their seminal treatment of games [58], von Neumann and Morgenstern explicitly forbid the possibility of infinite play. Although they also consider chess, they model it as a finite duration game by defining the play to immediately end in a draw whenever a position is repeated. Of course, if players are restricted to only use positional strategies, the two models coincide.

Payoff Models. The simplest way to handle infinite play is, like Zermelo, to combine all infinite plays into one outcome. Without loss of generality, this outcome is assumed to give payoff 0 to all players. This *terminal payoff* model is studied in Paper 1 for zero-sum games and in Paper 3 for multi-player games.

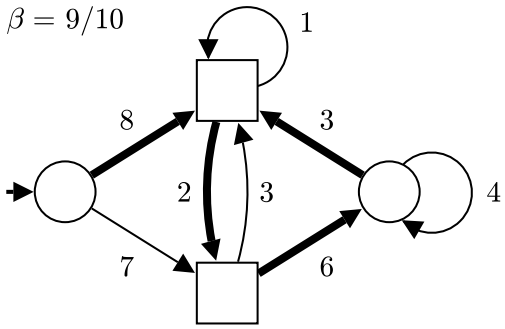


Figure 2.3: Zero-sum discounted-payoff game. The player to move is indicated by vertex shape: □ for Max and ○ for Min. The two positional strategies indicated in bold give Max the payoff $8+\frac{9}{10}\cdot 2+\frac{9^2}{10^2}\cdot 6+\frac{9^3}{10^3}\cdot 3+\frac{9^4}{10^4}\cdot 2+\ldots = 40\frac{175}{271}$.

A more general approach is to let payoffs accumulate throughout a play. By assigning payoffs to moves rather than terminals, each player receives an infinite stream $p_0, p_1, p_2, \ldots$ of intermediate payoffs. Terminal positions are typically disallowed in such models, but they can be simulated using "self-loops". We cannot define the total payoff p to a player simply as the sum of all his intermediate payoffs, since this sum does not necessarily converge. Sometimes, the limit average is used: $p := \liminf_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n-1} p_i$, also called the *mean payoff*. However, there is another type of "average" which turns out to be both mathematically and practically motivated: the *discounted* sum, i.e., $p := \sum_{i=0}^{\infty} \beta^i p_i$ for some β between 0 and 1. Figure 2.3 shows an example. There are at least three ways to interpret β: as a measure of inflation (later payoffs are less valued due to inflation), as a measure of interest (earlier payoffs can be invested to yield interest), or as an expectation in the presence of a stopping probability (after each move, with probability $1-\beta$ some event occurs that nullifies all later payoffs). Discounting can also be used in chess-like games, where it incentivizes the players to win as *quickly* as possible or postpone their loss for as long as possible [59].

A quite different method for evaluating infinite plays appears in so-called *parity games*. In these win/lose games, each position has an integer priority, and the winner is determined by the parity of the highest priority that appears infinitely often. Parity games and many other win/lose games on graphs arose in connection with temporal logics and model-checking problems [26].

Moves of Chance. A common extension to any game model is to add moves of chance, i.e., positions that do not belong to any of the players, but where "Nature" chooses a move at random. Thus, even when strategies are fixed, the play becomes a random object (in the case of positional strategies, a Markov chain on the positions). The payoffs are then naturally defined as expectations. Mean-payoff and discounted-payoff games with moves of chance can naturally be viewed as the competitive versions of infinite-horizon Markov decision processes [47]; see [18] for a treatment from this viewpoint.

Game models that allow for moves of chance are generally called *stochastic*

Table 2.1: Classification and overview of references to previous work on zero-sum perfect-information games on graphs.

	Terminal	Parity	Mean	Discounted
Deterministic	[59, 61]	[16, 29]	[2, 15]	[27, 63]
Stochastic	[9, 41]	[6, 7]	[25, 39]	[52]

games, with the others being *deterministic*. Some authors count Nature as half a player and refer to stochastic two-player games as "$2\frac{1}{2}$-player games" [7]. Several classes of zero-sum games with both cycles and moves of chance are studied in Paper 2.

2.3 Results

2.3.1 The Zero-Sum Case

The four different payoff models defined above together with the option of adding moves of chance, result in a total of eight distinct classes of games, which have all appeared in the literature under various names; see Table 2.1. The most basic question about a zero-sum game is whether it has a value. For all these eight classes, it turns out the answer is always affirmative and that something even stronger is true: Each player not only has an optimal strategy, but one that is both optimal and positional. In other words, all the games have positional Nash equilibria, but for zero-sum games this property is more often called *positional determinacy*.

Positional determinacy for *chess-like games*, i.e., deterministic terminal-payoff games with payoffs in $\{-1, 0, 1\}$, was established in a line of research started by Zermelo [61], whose arguments were later corrected and generalized by König [36] and Kalmár [32]; see Paper 1. A more advanced model was introduced in the seminal work of Shapley [52], who considered stochastic games with discounted payoffs.[1] Liggett and Lippman [39] established positional determinacy for stochastic mean-payoff games by correcting a proof by Gillette [25]. Their result also applies to terminal-payoff games via a trivial reduction. For deterministic and stochastic parity games, positional determinacy was proved by Emerson and Jutla [16] and Chatterjee et al. [7], respectively. A special case of stochastic terminal-payoff games was considered by Condon [9] under the name *simple stochastic games*. Ehrenfeucht and Mycielski [15] considered deterministic mean-payoff games.

In the light of these existence results, it is natural to consider the computational problem for each class: Given an explicitly represented game, compute its value and an optimal positional strategy for each player. We call this the problem of *solving* a game. Sometimes, we shall talk about the value of a particular position, which is simply the value of the (modified) game where that position is used as the starting position. For all the games here, it is in fact possible to

[1] Shapley's model actually allows for simultaneous moves. The positional determinacy of the perfect-information case that we consider here follows as a special case of his proof.

find positional strategies that are optimal regardless of which position is used as the starting position.

The folklore method of *retrograde analysis* is used in practice to solve chess endgames (where only a few pieces remain), but it can theoretically solve any chess-like game in full; see Paper 1 for historical details. Washburn showed that the generalization of chess-like games to arbitrary zero-sum payoffs, i.e., deterministic terminal-payoff games (by Washburn called *deterministic graphical games*), can also be solved in polynomial time. However, while retrograde analysis has a running time that is linear in the total size of the game graph (i.e., the number of positions and moves), Washburn's algorithm has a running time that is cubic in the number of positions. In Paper 1, we revisit Washburn's games to investigate whether more efficient algorithms can be constructed, with running times closer to that of retrograde analysis. We develop almost-linear time algorithms and study the comparison complexity of the problem, since the sorting of payoffs turns out to play a crucial role. We show that much sorting can be avoided by locating and focusing on the most relevant payoffs. On the other hand, we show that to find the values of *all* positions, a complete sorting is generally unavoidable. Although our algorithms constitute a significant improvement over Washburn's results, we leave the existence of linear time algorithms as an intriguing open problem.

For the seven remaining classes of games, it is unknown whether they can be solved in polynomial time. In several cases, randomized subexponential-time algorithms are known, starting with Ludwig [41] for simple stochastic games, whose ideas were adapted by Björklund and Vorobyov [2] for deterministic mean-payoff games, and put in a general framework by Halman [27]. For parity games, Jurdziński et al. [29] obtain a deterministic subexponential-time algorithm using a different approach, which does not easily generalize to the other classes. One of the most promising and widely applicable solution methods, the Hoffman-Karp algorithm, was recently shown to require exponential time in the worst case [22].

For all eight classes considered here, the associated decision problem "Given a game and a threshold t, is the value at least t?" is in both NP and coNP, since an optimal positional strategy for Player 1 or 2 shows that the answer is "yes" or "no", respectively (for all eight classes, the one-player version is solvable in polynomial time by, e.g., linear programming).

There are known polynomial-time reductions between several of these classes. In the deterministic case, solving parity games reduces to solving mean-payoff games [26], which in turn reduces to solving discounted-payoff games [63]. In Paper 2, we consider reductions between the stochastic classes. By generalizing results for deterministic games and combining them with some existing reductions, we manage to complete the picture: All four stochastic models are actually polynomial-time equivalent, i.e., if any one of them is solvable in polynomial time then all of them are. Whether any such equivalence holds between any of the deterministic game models is an open problem. For the sake of completeness, we untwine the tasks of computing values and constructing optimal strategies, and we also distinguish between unary and binary encodings of payoffs and probabilities. We show that all these variants are polynomial-

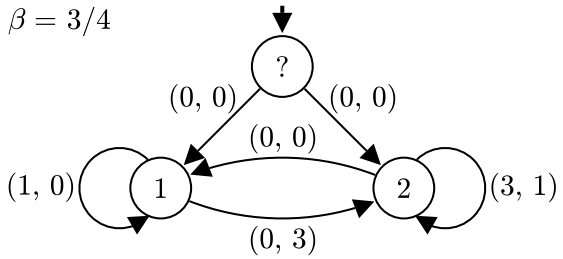


Figure 2.4: Non-zero-sum discounted-payoff game. The first move is chosen by a coin-toss ("?"). The unique stationary Nash equilibrium is for Player 1 and 2 to choose the self-loop with probability 1/3 and 7/12, respectively.

time equivalent. In contrast, unary *discount factors* enable polynomial-time algorithms [40].

2.3.2 The General Case

Non-zero-sum perfect-information games on graphs, with two or more players, have received less attention than their zero-sum and imperfect-information counterparts. Raghavan and Filar recognize this in their survey on stochastic games and suggest the perfect-information case as a promising area of study [48].

In this section we shall consider mixed strategies, i.e., strategies that use randomization. Such strategies did not show up in our discussion about the zero-sum games, because for those games we know that there are positional optimal strategies, which are also optimal against any mixed strategy. We will mainly be interested in a special type of mixed strategies. A *stationary strategy* (also *behavior strategy*) can be seen as a randomized version of (pure) positional strategies. Informally, a stationary strategy is memory-less, but it may use randomization. Thus, a stationary strategy for a player can be specified as a mapping from the positions of that player to probability distributions on the possible moves from each position. In our terminology, "positional" becomes synonymous with "stationary and pure". A stationary Nash equilibrium is one consisting only of stationary strategies.[2]

It follows from a general theorem by Fink [19] that all stochastic discounted-payoff games have stationary Nash equilibria. However, a simple example by Zinkevich et al. [62] shows, in our terminology, that such games do not always have *positional* Nash equilibria, i.e., the randomization in the stationary strategies of Fink is, in some sense, necessary; see Figure 2.4. It is interesting to note that randomization in strategies is never "necessary" for the acyclic games (they always have positional Nash equilibria). Thus, the introduction of cycles into a model can result in a need for randomization.

Thuijsman and Raghavan [55] consider stochastic mean-payoff games and show that although Nash equilibria can be constructed using pure history-

[2]Many authors instead define "stationary equilibrium" to be a profile of stationary strategies that is a Nash equilibrium regardless of which position is chosen as the starting position. This is called *subgame perfection* for game trees. We can simulate this stronger notion in a stochastic game by adding a starting position that randomly moves to any other position.

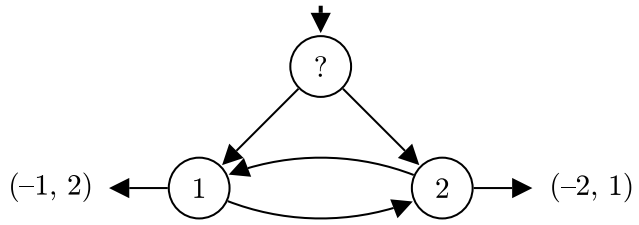


Figure 2.5: Non-zero-sum terminal-payoff game without stationary Nash equilibria. A (history-dependent) Nash equilibrium: Player 2 always chooses right, and Player 1 chooses left if he is the first player to move and right otherwise.

dependent strategies, stationary Nash equilibria do not generally exist, even for two players. A similar example, using only terminal payoffs, is shown in Figure 2.5. However, since these counterexamples use moves of chance, this leaves the door open for the deterministic case.

In Paper 3 we consider deterministic terminal-payoff games. Boros et al. [3, 4] have previously considered both more general and more special models. They established some conditions that that guarantee the existence of positional Nash equilibria (they used the term *Nash-solvability*), but they left the general case open.

A natural candidate method to search for positional Nash equilibria, if they exist, would be repeated strategy improvement: Starting from any positional strategy profile, repeatedly pick some player who can improve his strategy and update it, until either equilibrium is reached (no player can improve), or a profile is repeated (the search is cycling). In the latter case, we do not find an equilibrium, yet cannot conclude that one does not exist. However, if we could prove that no such "improvement cycles" exist, the existence of positional Nash equilibria would immediately follow.

Kukushkin [37,38] first considered this concept of "improvement acyclicity" in the context of game trees. He observed that if any type of improvement is allowed, even trees can have improvement cycles, but for more restricted forms of improvement, where changes are only allowed along the actual play, improvement acyclicity holds. In Paper 3, extending the work of Kukushkin, we investigate improvement acyclicity in games on graphs. We consider various types of restricted improvements (while maintaining the connection to positional Nash equilibria), and establish improvement acyclicity results for two cases: acyclic game graphs and two-player games. (Of course, for the acyclic case, positional Nash equilibria can easily be constructed using backward induction.) For two-player games, the existence of positional Nash equilibria was proved independently, using different methods, by Sørensen and Gimbert—their proof is included in Paper 3. We also establish negative results by exhibiting many examples of improvement cycles, showing the limitations of improvement cycles as a tool.

2.4 Features of Games on Graphs

In a broad sense, our line of theoretical work aims to investigate and lend insights into how different fundamental aspects of games, their "features", affect their feasibility from a computational perspective. Two examples of such features are the presence of cycles and moves of chance. Cycles seem to add considerable complexity; all the games under consideration here become easy to solve when their graph is acyclic. Regarding moves of chance, a dramatic example is chess-like games; they are solvable in linear time, but when moves of chance are added to the model, it is no longer known whether polynomial time algorithms exist. One way to interpret the equivalences of Paper 2 is to say that the presence of moves of chance seems more significant than the particular payoff model (with the caveat that all games considered therein might turn out to be polynomial-time solvable).

A particularly illustrative example of the intrinsic difficulty of dealing with cycles in games is given by the problem of *strategy recovery*. Consider a zero-sum terminal-payoff game with cycles. We cannot apply backward induction, since there is no "bottom up" ordering of the positions. But the difficulty runs deeper than that. Suppose we are generously given the value of every position in the game and are asked to construct an optimal strategy for Player 1. We call this the *strategy recovery* problem. We might try to solve this problem by simply making any locally optimal choice at each position—let Player 1 move to any successor with the same value (at least one must exist). While this would always work in the acyclic case, it does not generally work in the presence of cycles. To see this, consider a king & queen vs. king endgame in chess. Although every position is "won" for the queen-equipped player, not all his strategies will ensure that a checkmate is actually reached. In Papers 1 and 2 we show how strategy recovery can be performed in linear time for deterministic and stochastic terminal-payoff games, respectively. However, we do not know whether this problem can be solved even in polynomial time for the case of mean-payoff games—even deterministic ones.

We have chosen to focus on the high-level features of games, rather than their internal structure (e.g., how the chess-pieces move). Although we sometimes refer to popular games like chess, this should not be interpreted as suggesting that our work is motivated by practical application to these particular games—they simply serve as well-known and (mostly) well-defined examples used to illustrate concepts and classes of games. An alternative approach to the study of some chess-like games appears in the field of combinatorial game theory [1]. In that setting however, games are typically represented implicitly (by rules, rather than an explicit graph), and the focus is on finding closed form expressions that determine the winner, or more generally on developing an "arithmetic of games".

Chapter 3

NP-Complete Puzzles

3.1 Everyday Complexity

Every day, millions of people around the world encounter the puzzle section in their morning newspaper. For many of them, solving a Sudoku or some other pencil puzzle is their only hands-on experience with well-specified computational tasks more challenging than the basic arithmetic needed for everyday problems. Thus, a natural question for them to ponder—How hard could it be to solve Sudoku puzzles?—can serve as a connection between their experience and the field of computational complexity theory. Indeed, a wide range of different types of popular puzzles have been studied, analyzed and classified from a complexity-theoretic perspective; this tradition dates back to the early years of the field. Some surveys of the large collection of such results are given by Demaine and Hearn [13], Robertson and Munro [49], Eppstein [17], and Kendall et al. [34].

3.2 Formalizing Puzzles

In the context of computational complexity, a problem is actually an infinite family of instances. The question under consideration is how the resources needed to solve the problem increase with the size of the instances. This framework is not designed for making statements about the difficulty of solving any particular instance, since we are not considering any particular algorithm.[1] That we tend to agree, at least to some extent, on which instances of a puzzle are "hard" is likely due to many of us using similar methods for solving them. Complexity theory, on the other hand, enables us to make very general statements about what can be done by any algorithm, even those we did not think of yet.

Although many puzzle types are well-specified and sufficiently formal "out of the box", some require generalization or other modifications to fit the computational complexity framework. For example, a traditional Sudoku puzzle is based on a 9 by 9 grid to be filled with digits between 1 and 9, but we can easily

[1] However, see [5] for an attempt to formalize the intuitive notion of "hard" instances using Kolmogorov complexity.

consider the natural generalization with an n^2 by n^2 grid to be filled with numbers between 1 and n^2 [8,60]. Similarly, traditional Tangram puzzles are based on a fixed set of seven pieces, but we can consider arbitrary sets of polygonal pieces in general [45, page 211]. Sometimes, like in the case of crossword puzzles, only part of the problem can be meaningfully formalized—the challenge of figuring out the clues/hints does not easily lend itself to formalization, but the combinatorial aspect of fitting overlapping words together does [45, page 211].

Puzzles are in essence search problems, and the complexity class commonly used to capture search problems is NP [24]. However, the class NP is defined in terms of decision problems, i.e., problems where the output is either "yes" or "no". Informally, a decision problem is in NP if there is a short and easily verifiable proof that the answer is "yes" whenever this is actually the case. From this, it is immediate for most types of puzzles that the associated *solvability problem*—Given a puzzle instance, is there a solution?—is in NP; a solution, if one exists, is an easily verifiable proof. For many types of puzzles, it turns out that this problem is in fact NP-complete. Thus, it cannot be solved in polynomial time, unless all problems in NP can be solved in polynomial time. The classic NP-complete problem is the Boolean satisfiability problem: Given a Boolean formula, is there a satisfying truth assignment?

Paper 4 studies Hiroimono, a 600-year-old type of tour puzzle where stones must be collected from a grid [10]. We show that deciding the solvability of a given Hiroimono puzzle is NP-complete, using an elaborate reduction from the Boolean satisfiability problem. Paper 5 does the same for Hashiwokakero, a more recent puzzle from Japanese puzzle publisher Nikoli Co., where islands must be connected by building bridges. Therein, we reduce from an NP-complete special case of the Hamiltonian path problem.

Some types of puzzles do not "fit" in the class NP; this is typically due to some solutions requiring an exponentially long sequence of "steps", as in the case of Rush Hour [20] (getting out of a traffic jam) and Sokoban [11] (pushing boxes in a warehouse). For these puzzles, PSPACE—the class of problems solvable using a polynomial amount of memory—can be used instead of NP.

3.3 Realism and Relevance

At first glance, the solvability problem for puzzles may seem utterly unrelated to the task of a real life puzzle solver: the puzzles he encounters are always solvable! That is, the problem he solves is actually a so-called *promise problem*: Given a solvable puzzle, find a solution. However, it is clear that any efficient algorithm for this promise problem can also be used to decide the solvability: Given any instance (with unknown solvability status) just try the algorithm and see what it outputs (possibly an error message). If it actually outputs a solution, which we can check in polynomial time for NP problems, then the answer is "yes", otherwise it is "no". Thus, the promise of the existence of a solution does not significantly affect the applicability of our framework; we are still able to prove lower bounds on the complexity of the puzzle solving problem.

Another, more serious, possible objection to the "realism" of our approach

to puzzle complexity concerns *uniqueness* of solutions. Most real-life entertainment puzzles often come with a stronger promise: there is a unique solution. The theory for this kind of problem is not as developed as that of NP. Valiant and Vazirani [56] considered the unique solution version of the Boolean satisfiability problem. They proved that it cannot be solved in polynomial time unless all problems in NP can be solved by *randomized* polynomial-time algorithms (which can make random choices and are correct with high probability for any instance). To extend this result to other unique solution problems by means of reductions, we would have to use reductions that not only preserve solvability, but also the uniqueness of the solution, so-called parsimonious reductions.

One may also consider the corresponding problem faced by the puzzle constructor: Given a puzzle and a solution, decide if there are other solutions. This "another solution" type of problem has been shown to be NP-complete for several types of puzzles [60].

While the theory of unique solutions is interesting in its own right, its contribution to the realism of our modelling should not be overestimated. The primary reason for a magazine to reject the Hiroimono puzzles constructed by the reduction in Paper 4 would hardly be the fact that they may have multiple solutions, but rather that they are huge and look "strange".

Puzzles are interesting in that they are computational problems solved for entertainment. The complexity-theoretic study of popular puzzles does seem to indicate something about real life puzzle solving. Most "challenging" puzzles turn out to be NP-complete in their natural generalization. In contrast, many puzzles intended for younger children, like finding a path through a labyrinth, have simple polynomial time algorithms. Still, perhaps the greatest value of this line of research lies in its potential to popularize theoretical computer science in general, by drawing upon common experiences. The first NP-completeness results for Tetris [14] and Minesweeper [33] received mainstream media coverage by CNN and the BBC.[2] Interestingly, the formalizations used in these results are arguably quite unrealistic in several ways, but they nevertheless succeeded in raising interest and curiosity among people without any prior contact with theoretical computer science.

[2]Somewhat less impressively, Paper 4 was referenced in the puzzle blog of Polish news magazine Polityka.

Chapter 4

Local Network Interdiction

4.1 Network Interdiction

Networks are ubiquitous. Some are constructed for the purpose of transportation and communication, while others, less tangible, arise from our personal ties in the form of social networks. Consequently, much effort has been spent on methods for designing and optimizing the use and functionality of networks. Finding shortest paths, minimum spanning trees, and maximum flows, are all enormously applicable problems, for which an extensive body of theory and algorithms has been developed.

The simplest models used for optimization, like the shortest path problem, assume that parameters, like edge weights, are fully known in advance, and that all components of the network work reliably and consistently. *Network interdiction* problems concern design and optimization on networks in the presence of disruption or destruction, like the removal of edges or vertices. Here, we will focus on the case where the cause of this disruption is an intentionally malicious adversary, giving the problem a game-theoretic flavour. However, the term "network interdiction" is sometimes also used for models with stochastic or other non-strategic causes such as natural disasters, wear and tear, or human error. Figure 4.1 shows a simple example of how disruption can affect network usability.

4.2 History and Models

The study of network interdiction problems can be traced back at least to the work by Ford and Fulkerson on the max-flow/min-cut problem, which was motivated by interdiction of enemy railway networks [51]. In a network with source s, sink t and unit capacities, the minimum cut problem simply asks how many edges need to be removed to disconnect s from t. With arbitrary capacities, we can consider a model where capacities are decreased: Given a "budget" of total capacity that can be removed, choose where to spend this budget to minimize the resulting maximum flow. Again, the optimal choice is to reduce capacities across the minimum cut.

The commonly used model is a two-stage perfect-information game: The

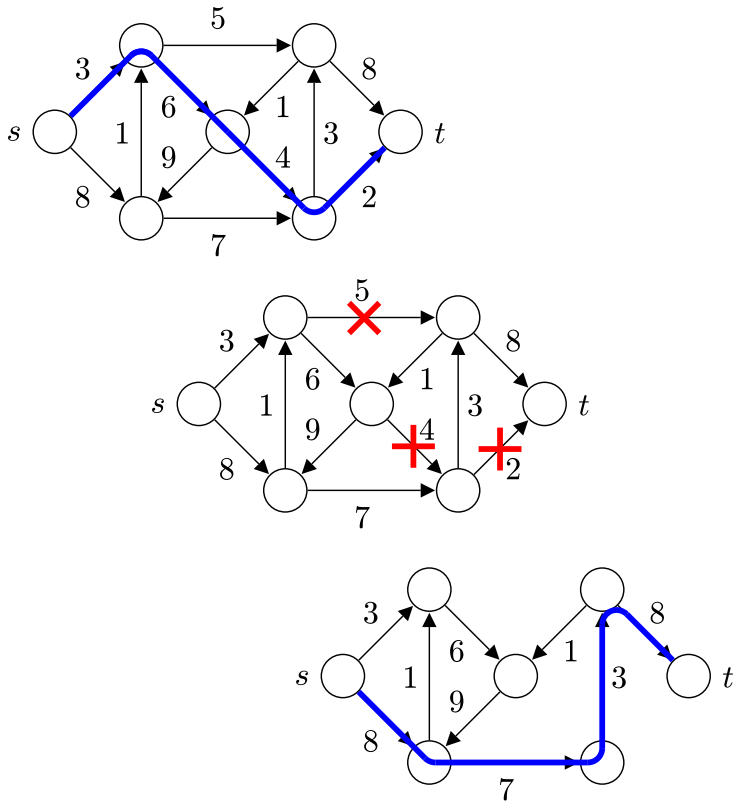


Figure 4.1: An example of network interdiction. Removing the crossed edges increases the length of the shortest path from 21 to 26.

first player, called Interdictor, modifies the network subject to some given budget constraints, and then the second player, called User or Operator, performs some optimization on the modified network. Interdictor seeks to maximize the minimum cost, or equivalently, minimize the maximum utility, for User. This can be seen as a zero-sum case of the classic Stackelberg leader/follower competition model [54].

Network interdiction problems have received considerable attention in the military operations research community. The seminal work of McMasters and Mustin [42], and later, Philips [46], considered minimizing the maximum flow. Another common model is the maximization of the shortest path; see Israely and Wood [28]. For the models used in applied operations research, the solution method typically involves modelling the problem as a mixed integer linear program, which is then solved using specially tailored heuristics. The worst case running time of the algorithms is typically super-polynomial. For those models, this is most likely unavoidable, since the problems tend to be NP-hard even in a simple form. For instance, the following simple version of shortest path interdiction is NP-hard [50]: Given an s-t network, a budget k, and a target l, decide whether Interdictor can remove k edges so that the length of the shortest s-t path becomes at least l. Furthermore, Khachiyan et al. [35] show that the length l is NP-hard to approximate within a factor less than 2.

In Paper 6, following Khachiyan et al., we adopt an algorithmic viewpoint that asks "What models are amenable to worst-case efficient algorithms?".

4.3 Local Budgets

Khachiyan et al. [35] considered a version of shortest path interdiction on weighted directed graphs where the budgets of Interdictor are *local* to each vertex. That is, for each vertex v, Interdictor has a separate budget restricting how outgoing edges from v may be removed. A simple type of budget would be a number $k(v)$ for each vertex, specifying how many outgoing edges may be removed, but Khachiyan et al. also considered the more general case with each budget specified as a family of all "affordable" subsets of the outgoing edges. They showed that with such local budgets, optimal choices for Interdictor can be computed in polynomial time using a modified Dijkstra's algorithm.

In [35], the following example is used as an illustration: Each edge (u, v) has an associated probability $p(u, v)$ that contraband smuggling from city u to city v can be carried out undetected. At each vertex, the local authorities can choose some outgoing connections for intensive monitoring that catches all smuggling (arguably, the local authorities might be more interested i monitoring *incoming* edges instead of outgoing, but the model can easily be changed accordingly). If the local authorities are working together (but with separate budgets) towards the common goal of preventing smuggling between a source and a sink, then we can model this as a shortest path interdiction problem with vertex-local budgets by defining the weight of an edge (u, v) as $-\log p(u, v)$.

In Paper 6 we extend the model of Khachiyan et al. in two directions. First, we add the capability of *partial interdiction*, i.e., instead of just removing edges, Interdictor can choose how much to increase their weight (in the example, decrease the probability of successful smuggling). Also, we consider more general ways for User to evaluate his paths. In the above example, there was a simple transformation from probabilities to additive weights, but this may not always be possible. Our extended model can also be efficiently solved using a modified Dijkstra's algorithm.

An interesting problem for future research is to investigate whether other greedy graph algorithms, such as Prim's algorithm for constructing minimum cost spanning trees, can be similarly extended to handle some form of local interdiction. Also for minimum spanning trees, the global budget version is known to be NP-hard [21].

4.4 Connections to Games on Graphs

As noted in [35], there are connections between shortest path interdiction problems and some of the zero-sum games on graphs described in Chapter 2. Indeed, in both cases are the two players influencing the formation of a path. If at each vertex, Interdictor may either remove no edges or all but one edge (i.e., selecting which one to keep), the correspondence becomes even more clear—each node is completely controlled by either User or Interdictor.

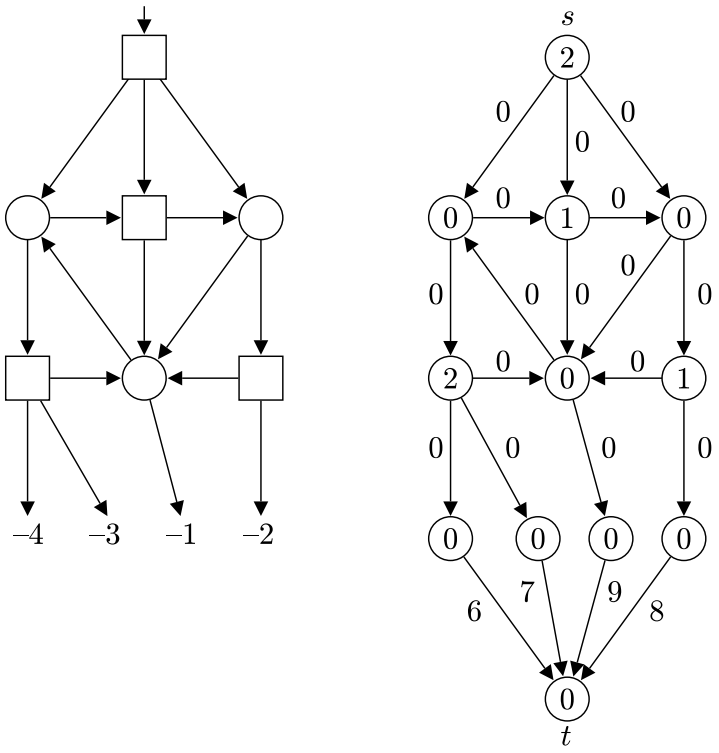


Figure 4.2: Reducing a (negative) deterministic graphical game (*left*) to a shortest path interdiction problem with local budgets (*right*). On the left, the player to move is indicated by vertex shape: □ for Max and ○ for Min. On the right, the number in each vertex is the number of outgoing edges Interdictor may remove.

Figure 4.2 shows a deterministic graphical game and an equivalent shortest path interdiction problem. Forming a cycle (i.e., infinite play) in the game, corresponds to Interdictor disconnecting s from t in the interdiction instance—his most preferred outcome. Therefore, this immediate reduction only works when all terminal payoffs in the game are negative. In the reduction, these payoffs are all shifted by the same amount (10 in Figure 4.2).

However, Paper 1 shows how the problem of solving any deterministic graphical game can be reduced, in linear time, to solving games where all terminal payoffs have the same sign. Thus, we obtain a linear-time reduction also from the general case.

Bottlenecks. Note that the s-t paths in the network produced by the above reduction only have a single non-zero weight. Thus, instead of minimizing total weight, User would get the same result if he wanted a *bottleneck shortest path*—one that minimizes the *heaviest* edge along the path.

Bottleneck variants have been considered for many classic optimization problems on graphs [23], and often allow faster algorithms than their "total sum" counterparts. This turns out to be the case also for interdiction with local budgets. Paper 6 shows how the techniques of Gabow and Tarjan [23] can

be applied to yield an almost-linear time algorithm for bottleneck interdiction with local budgets of edge removals. The time bound matches the best known bound for the standard bottleneck shortest path problem in directed graphs (for undirected graphs, linear time algorithms are known [31]).

Bibliography

[1] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for your Mathematical Plays*. Academic Press, 1982.

[2] H. Björklund and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155(2):210–229, 2007.

[3] E. Boros and V. Gurvich. On Nash-solvability in pure strategies of finite games with perfect information which may have cycles. *Math. Social Sciences*, 46, 2003.

[4] E. Boros, V. Gurvich, K. Makino, and W. Shao. Nash-solvable bidirected cyclic two-person game forms. Technical Report 2008-13, DIMACS, Rutgers University, 2008.

[5] H. Buhrman and P. Orponen. Random strings make hard instances. *J. Comput. Syst. Sci.*, 53(2):261–266, 1996.

[6] K. Chatterjee and T. A. Henzinger. Reduction of stochastic parity to stochastic mean-payoff games. *Inf. Process. Lett.*, 106(1):1–7, 2008.

[7] K. Chatterjee, M. Jurdziński, and T. A. Henzinger. Quantitative stochastic parity games. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 121–130, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[8] C. J. Colbourn. The complexity of completing partial latin squares. *Discrete Applied Mathematics*, 8(1):25 – 30, 1984.

[9] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[10] M. J. Costello. *The greatest puzzles of all time*. Prentice Hall Press, 1988.

[11] J. Culberson. Sokoban is PSPACE-complete. In *Proceedings of the International Conference on Fun with Algorithms*, pages 65–76. Carleton Scientific, 1998.

[12] C. Daskalakis, R. M. Karp, E. Mossel, S. Riesenfeld, and E. Verbin. Sorting and selection in posets. In *SODA '09: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 392–401,

Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[13] E. D. Demaine and R. A. Hearn. Playing games with algorithms: Algorithmic combinatorial game theory. In M. H. Albert and R. J. Nowakowski, editors, *Games of No Chance 3*, volume 56 of *Mathematical Sciences Research Institute Publications*, pages 3–56. Cambridge University Press, 2009.

[14] E. D. Demaine, S. Hohenberger, and D. Liben-Nowell. Tetris is hard, even to approximate. In *Proceedings of the 9th Annual International Conference on Computing and Combinatorics*, volume 2697 of *Lecture Notes in Computer Science*, pages 351–363. Springer-Verlag, 2003.

[15] A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8(2):109–113, 1979.

[16] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS 1991)*, pages 368–377. IEEE, 1991.

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

[18] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1997.

[19] A. M. Fink. Equilibrium in a stochastic n-person game. *J. Sci. Hiroshima Univ. Ser. A-I Math.*, 28:89–93, 1964.

[20] G. W. Flake and E. B. Baum. Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants". *Theor. Comput. Sci.*, 270(1–2):895–911, 2002.

[21] G. N. Frederickson and R. Solis-Oba. Increasing the weight of minimum spanning trees. In *SODA '96: Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 539–546, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.

[22] O. Friedmann. A super-polynomial lower bound for the parity game strategy improvement algorithm as we know it. In *Proceedings of the Twenty-Fourth Annual IEEE Symposium on Logic in Computer Science (LICS 2009)*, 2009. (to appear).

[23] H. Gabow and R. Tarjan. Algorithms for two bottleneck optimization problems. *J. Algorithms*, 9:411–417, 1988.

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

[25] D. Gillette. Stochastic games with zero stop probabilities. In M. Dresher, A. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games III*, volume 39 of *Annals of Mathematics Studies*, pages 179–187. Princeton University Press, 1957.

[26] E. Grädel and W. Thomas. *Automata, Logics, and Infinite Games: A Guide to Current Research*. Springer, 2002.

[27] N. Halman. Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems. *Algorithmica*, 49(1):37–50, 2007.

[28] E. Israely and K. Wood. Shortest-path network interdiction. *Networks*, 40:97–111, 2002.

[29] M. Jurdziński, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM J. Comput.*, 38(4):1519–1532, 2008.

[30] J. Kahn and J. H. Kim. Entropy and sorting. *Journal of Computer and System Sciences*, 51(3):390–399, 1995.

[31] V. Kaibel and M. A. Peinhardt. On the bottleneck shortest path problem. Technical Report ZR-06-22, Konrad-Zuse-Zentrum fur Informationstechnik Berlin, 2006.

[32] L. Kalmár. Zur Theorie der abstrakten Spiele. *Acta Sci. Math. Szeged*, 4:65–85, 1928.

[33] R. Kaye. Minesweeper is NP-complete. *Mathematical Intelligencer*, 22:9–15, 2000.

[34] G. Kendall, A. Parkes, and K. Spoerer. A survey of NP-complete puzzles. *International Computer Games Association Journal*, 31(1):13–34, 2008.

[35] L. Khachiyan, E. Boros, K. Borys, K. M. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233, 2008.

[36] D. König. Über eine Schlussweise aus dem Endlichen ins Unendliche. *Acta Sci. Math. Szeged*, 3:121–130, 1927.

[37] N. Kukushkin. Perfect information and potential games. *Games and Economic Behavior*, 38:306–317, 2002.

[38] N. Kukushkin. Acyclicity of improvements in finite game forms. `http://www.ccas.ru/mmes/mmeda/ququ/GF.pdf`, 2009.

[39] T. M. Liggett and S. A. Lippman. Stochastic games with perfect information and time average payoff. *SIAM Review*, 11(4):604–607, 1969.

[40] M. L. Littman. *Algorithms for sequential decision making*. PhD thesis, Brown University, Department of Computer Science, 1996.

[41] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117(1):151–155, 1995.

[42] A. McMasters and T. Mustin. Optimal interdiction of a supply network. *Naval Research Logistics Quarterly*, 17:261–268, 1970.

[43] J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950.

[44] B. O'Neill. A survey of game theory studies of peace and war. In R. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, volume 2. Elsevier, 1994.

[45] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[46] C. Philips. The network inhibition problem. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 776–785, 1993.

[47] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley-Interscience, 1994.

[48] T. Raghavan and J. Filar. Algorithms for stochastic games—a survey. *Mathematical Methods of Operations Research*, 35(6):437–472, 1991.

[49] E. Robertson and I. Munro. NP-completeness, puzzles and games. *Utilitas Math.*, 13:99–116, 1978.

[50] B. Schieber, A. Bar-Noy, and S. Khuller. The complexity of finding most vital arcs and nodes. Technical Report UMIACS-TR-95-96, University of Maryland Institute for Advanced Computer Studies, College Park, MD, USA, 1995.

[51] A. Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2002.

[52] L. Shapley. Stochastic games. *Proc. Nat. Acad. Science*, 39:1095–1100, 1953.

[53] J. M. Smith. *Evolution and the theory of games.* Cambridge University Press, 1982.

[54] H. Stackelberg. *Marktform und Gleichgewicht.* Julius Springer, 1934.

[55] F. Thuijsman and T. Raghavan. Perfect information stochastic games and related classes. *International Journal of Game Theory*, 26:403–408, 1997.

[56] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986.

[57] J. von Neumann. Zur theorie der gesellschaftsspiele. *Math. Annalen*, 100:295–320, 1928.

[58] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press, Princeton, 1953. 3rd edition.

[59] A. Washburn. Deterministic graphical games. *Journal of Mathematical Analysis and Applications*, 153:84–96, 1990.

[60] T. Yato and T. 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.

[61] E. Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proceedings of the Fifth International Congress of Mathematicians*, pages 501–504. Cambridge University Press, 1913.

[62] M. Zinkevich, A. Greenwald, and M. Littman. Cyclic equilibria in Markov games. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1641–1648. MIT Press, Cambridge, MA, 2006.

[63] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1-2):343–359, 1996.

Part II

Papers

Paper 1

Deterministic Graphical Games Revisited

Daniel Andersson, Kristoffer Arnsfelt Hansen
Peter Bro Miltersen, Troels Bjerre Sørensen

Abstract. Starting from Zermelo's classical formal treatment of chess, we trace through history the analysis of two-player win/lose/draw games with perfect information and potentially infinite play. Such *chess-like* games have appeared in many different research communities, and methods for solving them, such as *retrograde analysis*, have been rediscovered independently.

We then revisit Washburn's *deterministic graphical games*, a natural generalization of chess-like games to arbitrary zero-sum payoffs. We study the complexity of solving deterministic graphical games and obtain an almost-linear time comparison-based algorithm for finding optimal strategies in such games. The existence of a linear time comparison-based algorithm remains an open problem.

This paper is to appear in *Journal of Logic and Computation*. A shorter version appeared in *Proceedings of the Fourth Conference on Computability in Europe* (CiE 2008), LNCS 5028.

1 Background

The most natural thing to do in the presence of potentially infinite play seems to be to simply permit it...

—Alan Washburn [39]

1.1 Zermelo's Treatment of Chess

In 1913, Zermelo published his classical treatment of chess [40, 32]. In modern terms, Zermelo modelled chess using a finite directed graph, whose vertices are the possible positions of the game, and whose arcs correspond to possible moves. Here, a position means not only the configuration of the board, but also which player is to move (and other state information regarding castling, *en passant* moves etc.). Positions where play ends, i.e., *terminal* positions, are the sinks of the graph. Each such position is labelled with $1, -1$, or 0, according as the position is a win for White, a win for Black, or a draw (stalemate), respectively. This number is interpreted as a payment from Black to White, also known as White's *payoff*. White seeks to maximize this payoff, while Black seeks to minimize it. In game-theoretic terms, chess is a *zero-sum* game. Accordingly, we will also use the names *Max* and *Min* for White and Black, respectively. The non-terminal positions, where a move (i.e., an outgoing arc) must be selected, are partitioned between the two players, according as who is to move. The players construct a path, starting from the designated starting position and consisting of the arcs selected during the play. It is possible that the play never ends, in which case the outcome is defined to be a draw, i.e., the payoff is 0. Infinite play is also consistent with modern FIDE rules; the "50-move" rule and the "threefold repetition" rule may permit a player to request a draw, but they do not automatically terminate play. Following Ewerhart [11], we will call games of this general form *chess-like* games (the game of chess as considered by Zermelo corresponding to one particular graph).

Zermelo's main motivation was to give mathematically rigorous definitions of the concepts now known as "values" of positions and "optimal" strategies. Zermelo set out to prove the following theorem: For each position, either White can force a win (in modern terminology, defined formally below, the *lower value* of the position is 1), or Black can force a win (the *upper value* of the position is -1), or neither player can force a win, i.e., each player can force a draw or better (both the lower and the upper value of the position is 0).

A crucial lemma used in his attempted proof was the following: If a player can force a win, then he can guarantee this to occur within a number of moves less than the total number of positions. In his proof of this lemma, Zermelo implicitly assumes that if a position is visited twice during a play, the opponent will choose the same move the second time. Thus, he fails to take into account the possibility of history-dependent behavior. This flaw was later pointed out by König, and a correct argument was given in [24], completing the proof of Zermelo's theorem.

Strategies and Values. While Zermelo's theorem established the existence of values, the statement does not provide much information about the nature of optimal strategies. In fact, Zermelo and König did not formally define strategies, making it difficult to argue correctly about "optimal play" and contributing to later confusion regarding what was actually proved by Zermelo [32, 11]. A rigorous definition of strategies was contributed by Kalmár [18] in a more general setting including all chess-like games. A *strategy* for a player is any plan that specifies how he will choose the outgoing arc whenever he is to move. In the modern understanding of the general term "strategy", such a choice may depend on the history of play (i.e., how the current position was reached) and may use randomization. Strategies that do not use any of these two features are called *pure positional* strategies. Thus, a pure positional strategy for a player is simply a selection of one fixed arc for each of his positions.

The *lower value* of a position q is the supremum of numbers L such that Max has a strategy that ensures a payoff of at least L against any counter-strategy of Min when play starts from q. Similarly, the *upper value* of a position q is the infimum of numbers U such that Min has a strategy that ensures a payoff of at most U against any counter-strategy of Max when play starts from q. When the upper and lower values are equal, this quantity is simply called the *value* of the position. With this terminology, Zermelo's theorem can be rephrased: "Each position in a chess-like game has a value".

A strategy for Max (Min) is called *optimal for a position* q if it guarantees a payoff of at least (at most) the lower (upper) value of q against any counter-strategy when play starts from q. Kalmár showed that each player has single pure positional strategy that is optimal for *all* positions. A strategy with this property is simply called *optimal*.

More extensive surveys of these early works of Zermelo, König and Kalmár are given by Schwalbe and Walker [32] and Ewerhart [11]. Neither the original works, nor these surveys, consider the problem of efficiently constructing optimal strategies or computing values. This is the main concern of this paper.

Weak and Strong Solutions. In our computational setting, we will make the following distinction.

- A *weak solution* is the value of the specified starting position v_0 and for each player a pure positional strategy that is optimal for v_0.
- A *strong solution* is a list of the values of all positions and for each player a pure positional strategy that is optimal for all positions.

In this paper, we consider the computational problem of finding a weak or strong solution to a game given explicitly as a graph.

The difference between weak and strong solutions is analogous to the difference between the notions of Nash equilibria and subgame-perfect equilibria defined for finite extensive-form games (see, e.g., [37]). In fact, when the game graph of a chess-like game is a tree, this correspondence becomes completely rigorous: The weak solutions of the chess-like game are exactly the Nash equilibria of the extensive form game defined by that tree, and the strong solutions

are exactly the subgame-perfect equilibria. Figure 1 illustrates that the distinction is not inconsequential. In the weak solution on the left, Max (if he gets

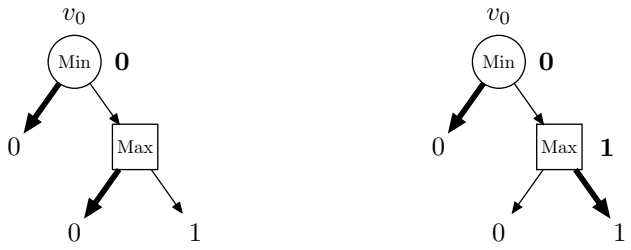


Figure 1: The left solution is weak, the right is strong.

to move) is content to achieve payoff 0, the value of the game, even though he could achieve payoff 1.

We shall say that we weakly (resp. strongly) solve a given game when we compute a weak (resp. strong) solution to the game. Note that when talking about strong solutions, the starting position is irrelevant and does not have to be specified.

Some simple examples of chess-like games are given in Figure 2. In (a), the unique strong solution is for Min to choose right and for Max to choose left. Thus, the outcome is infinite play. In (b), the unique strong solution is for Min to choose right and for Max to choose right. The values of both positions are 1, but we observe that it is *not* a sufficient criterion for optimal play to choose a position with at least as good a value as your current position. In particular, according to this criterion, Max could always choose left, but this would lead to infinite play and a payoff of 0, which is a suboptimal outcome for Max.

1.2 Algorithms Solving Chess-Like Games

It is a common misconception [32, 11] that Zermelo used the method of *backward induction* to solve chess. This method computes values in a "bottom-up" fashion, starting from the terminals and ensuring that any position is evaluated before its predecessors. Clearly, this is only possible for acyclic game graphs. Backward induction was first given formal treatment by von Neumann and Morgenstern [38] in 1944. They explicitly disallow the possibility of infinite play in their treatment of games [38, page 58].

Nevertheless, it is possible to reduce any chess-like game to a finite duration game, which can then be solved using backward induction: The positions of

(a) (b)

Figure 2: (a) Infinite play (b) All values are 1, but not all strategies are optimal

the finite game are all those partial plays of the original game that do not contain repeated positions or end at their first repeated position. Arcs are defined in the obvious way. Hence, there are two types of terminal positions: those partial plays actually being complete plays in the original game—these are assigned the corresponding payoffs—and those partial plays ending at their first repeated position—these are assigned the payoff zero. It is easy to verify that the initial position of the game thus constructed has the same value as the initial position of the original game. Note that this reduction blows up the size of the game graph exponentially.

The backward induction algorithm has been the basis of a range of more refined algorithms that evaluate typical acyclic games using time sublinear in the size of the game graph (the number of positions and arcs). Sublinear time is relevant when the game graph is represented implicitly (by the rules of the game) rather than explicitly, and this is often the case in the practical applications. Examples of such possibly sublinear-time algorithms include α-β pruning [23] and A* [16]. The challenge of generalizing these techniques to game graphs with cycles has been called "the graph history interaction problem" in the AI literature [28, 20]. This challenge has been addressed to the extent that the value of, e.g., Checkers has been determined [31]—it is a draw. However, our main focus in this paper is on the case of explicitly represented game graphs and worst case complexity bounds. In this setting, a different bottom-up approach is needed to handle games with potentially infinite play: *retrograde analysis*.

Retrograde analysis works by successively labelling positions with their values and coloring arcs green ("optimal") or red ("redundant"). The green arcs will form the optimal strategies. Initially, only terminal positions are labelled with their values and no arcs are colored. Then, we repeatedly pick an uncolored arc (u, v) with u unlabelled and v labelled with a non-zero label. Without loss of generality, assume that this label is 1 (the case of -1 is handled symmetrically).

- If u belongs to Max, we color (u, v) green and label u with 1.
- If u belongs to Min, there are two cases: If (u, v) is the only uncolored outgoing arc from u, then we color (u, v) green and label u with 1. If there are other uncolored outgoing arcs from u, then we color (u, v) red.

When, eventually, no such arc (u, v) exists, then for each unlabelled position, we arbitrarily color one of its uncolored outgoing arcs green and label the position with 0.

The correctness of this algorithm can be established by a proof by induction that the following three-part invariant is maintained.

1. Each non-terminal labelled position has exactly one outgoing green arc, and each unlabelled position has at least one outgoing non-red arc.
2. For any red arc (u, v), the position v is labelled and the owner of u prefers 0 to the label of v.
3. For any player, consider a strategy that never uses red arcs and always uses the green arc at labelled positions. For each labelled position u, such

> a strategy guarantees the player a payoff at least as good as the label of u when play starts from u.

Like backward induction, retrograde analysis can be performed in time linear in the size of the game graph.

The above argument not only proves that the algorithm correctly and efficiently computes values and optimal strategies, but it also proves that these objects *exist*. We want to emphasize that this is arguably a simpler proof than that of Kalmár (see, e.g., the exposition by Ewerhart [11]).

The term retrograde analysis originated in the context of a type of chess puzzle where one must determine which moves could lead to a given (seemingly inexplicable) position. Such a puzzle is called a *retro*. Of course, the entire above algorithm is not used in this case, only the basic operation of following an incoming arc to a given position.

While it is theoretically possible to solve the entire game of chess using retrograde analysis, the game graph is much too large for this to be practical. However, many endgames of chess, where only a few pieces remain, have been completely solved using computers. The first published example is from 1970 by Ströhlein [35] (see also [27]). As computing power increased, more and more interesting endgames were solved [36, 34]. Later findings have contradicted many conventional beliefs based on experience and heuristics [34].

Ströhlein references Knuth [21, page 270] for the method used to solve the endgames. This reference is to an exercise wherein Knuth describes the principle of retrograde analysis (without using that term) but leaves it to the reader to devise an efficient implementation (and prove its correctness). Stiller [34] credits Bellman [4] as the first to propose the use of retrograde analysis by computers to solve chess endgames. While Bellman does discuss solving chess endgames by dynamic programming, in particular the possible benefits of decomposing game graphs into their strongly connected components, he does not actually describe the necessary retrograde analysis step, nor does he mention that term or provide any reference. Thus, it appears that both Knuth and Bellman considered the basic operation of retrograde analysis to be part of folklore. Several later authors outside the chess solving community have described retrograde analysis without reference to that name or to any previous appearances. For instance, Immerman [17] uses retrograde analysis to solve the "alternating graph reachability problem", and Condon [9] uses it to solve a chess-like special case of her more general "simple stochastic games".

1.3 Deterministic Graphical Games

Washburn [39] considered a natural generalization of chess-like games, where terminal positions may have arbitrary zero-sum payoffs. Infinite play is still interpreted as a zero payoff. Washburn named this class *deterministic graphical games*. Thus, chess-like games are deterministic graphical games with payoffs from the set $\{-1, 0, 1\}$. The term "graphical" refers only to the fact that the positions and moves can be seen as the vertices and arcs of a directed graph, possibly containing cycles, in contrast with the trees that are commonly used to

model games of finite duration. There is no relation to a more recent use of the term "graphical games" that refers to a succinct representation for multi-player games [19].

The definitions of values and optimal strategies presented in Section 1.1 generalize verbatim to the case of deterministic graphical games. However, it is not clear, *a priori*, that the corresponding existence results similarly generalize. To establish such existence results, Washburn recognizes a link between his simple extension of chess-like games and the vast literature on so-called stochastic games originating in Shapley [33]. He attributes the proofs of existence of values and optimal positional strategies to Everett [10]. In fact, this attribution is arguably incorrect. Everett considers a more general class of games, and while the games in this class do have values, these values cannot be strategically guaranteed by the players, but only approximated arbitrarily well. However, the existence of optimal strategies for deterministic graphical games does follow from a general theorem of Liggett and Lippman [25], who corrected a proof of Gillette [13].

Washburn [39] gave an algorithm for computing a strong solution to a deterministic graphical game. Washburn's algorithm contains some elements of retrograde analysis, but its running time is cubic in the number of positions. The goal of the second part of this paper is to construct more efficient algorithms, with time bounds closer to those achievable for chess-like games.

In [39], Washburn also considered a *discounted* version of deterministic graphical games, where the payoff at a terminal depends on the number of steps taken to reach it. This was later generalized by Baston and Bostock [3]. We will not consider such variants here.

Interesting natural examples of deterministic graphical games have been solved by computers in practice. Awari is a game in the mancala family, where the objective is to capture more stones than the opponent. It can be modelled as a deterministic graphical game with payoffs in the range $[-48, 48]$. Romein and Bal [30] strongly solve awari—it is a draw. However, to reduce the number of states, they do not model it as a deterministic graphical game; they do not include the current capture balance as part of the state, so payoffs are instead accumulated throughout a play. They then make several passes of a variant of retrograde analysis through the entire database, one for each possible payoff. We note that their method leads to an algorithm for strongly solving deterministic graphical games in $\Theta(mn)$ time, where m is the number of arcs and n is the number of terminals.

1.4 Other Models

The study of more general two-player zero-sum games of infinite duration has generated considerable interest. Within the computer science community, much of the interest is due to their applicability to model-checking [14]. The simple game models presented herein can be augmented in several ways. One can introduce moves of chance, e.g., positions where a successor is chosen by a cointoss, as in Condon's *simple stochastic games* [9]. Another extension is to assign payoffs to non-terminal positions, and have payoffs accumulate throughout an

infinite play, either as a limit average or a discounted average [41]. In both of these models, the games possess optimal pure positional strategies, but it is a long-standing open problem whether they can be solved in polynomial time, although several subexponential algorithms are known [26, 15, 6] and many equivalences have been established [25, 41, 8, 2].

Chess-like games share some features with the *loopy games* studied in the field of combinatorial game theory [5]. However, in that setting, the focus is on developing a theory for the composition of many games into one—a calculus of games. This was originally inspired by the endgames of Go, where several non-interacting parts may appear.

2 New Results

We consider the computational problems of finding a weak or a strong solution to a given deterministic graphical game.

We observe below that if a list with the terminals ordered by payoff is given as additional input, optimal strategies can be found in linear time, as in the case of retrograde analysis. From this it follows that a deterministic graphical game with n terminals and m arcs in the graph can be strongly solved in time $O(n \log n + m)$ by a comparison-based algorithm. The main question we approach in this paper is the following:

> **Main Question.** *Can a deterministic graphical game be (weakly or strongly) solved in linear time by a comparison-based algorithm?*

We believe this to be an interesting question, both in the context of solving infinite duration games (deterministic graphical games being a very simple yet non-trivial natural variant of the general problem) and in the context of the study of comparison-based algorithms and comparison complexity. This paper provides neither a positive nor a negative answer to the question, but we obtain a number of partial results, described below.

Throughout this paper we consider deterministic graphical games with n denoting the number of terminals (i.e., number of payoffs) and m denoting the total size (i.e., number of arcs) of the graph defining the game. We can assume $m \geq n$, as terminals without incoming arcs are irrelevant.

Strategy Recovery in Linear Time. The example of Figure 2(b) shows that it is not completely trivial to obtain a strong solution from a list of values of the positions. We show that this task can be done in linear time, i.e., time $O(m)$. Thus, when constructing algorithms for obtaining a strong solution, one can concentrate on the task of computing the values $\mathrm{Val}(u)$ for all u. Similarly, we show that given the value of just the starting position, a weak solution to the game can be computed in linear time.

The Number of Comparisons. When considering comparison-based algorithms, it is natural to study the number of comparisons separately from the running time of the algorithm. By an easy reduction from sorting, we show that there is no comparison-based algorithm that *strongly* solves a given game using only $O(n)$ comparisons. In fact, $\Omega(n \log n)$ comparisons are necessary.

In contrast, Mike Paterson (personal communication) has observed that a deterministic graphical game can be *weakly* solved using $O(n)$ comparisons and $O(m \log n)$ time. With his kind permission, his algorithm is included in this paper. This also means that for the case of weak solutions, our main open problem cannot be resolved in the negative using current lower-bound techniques, as it is not the number of comparisons that is the bottleneck.

Our lower bound for strong solutions uses a game with $m = \Theta(n \log n)$ arcs. Thus, the following interesting open question concerning only the comparison complexity remains: *Can a deterministic graphical game be* strongly *solved using $O(m)$ comparisons?* If the answer turns out to be "no", it would resolve our main open problem for the case of strong solutions.

Almost-Linear Time Algorithm for Weak Solutions. As stated above, Mike Paterson has observed that a deterministic graphical game can be weakly solved using $O(n)$ comparisons and $O(m \log n)$ time. We refine his algorithm to obtain an algorithm that weakly solves a game using $O(n)$ comparisons and only $O(m \log \log n)$ time. Also, we obtain an algorithm that weakly solves a game in time $O(m + m(\log^* m - \log^* \frac{m}{n}))$ but uses a superlinear number of comparisons. For the case of *strongly* solving a game, we have no better bounds than those derived from the simple algorithm described in Section 3, i.e., $O(m + n \log n)$ time and $O(n \log n)$ comparisons. Note that the bound $O(m + m(\log^* m - \log^* \frac{m}{n}))$ is in fact linear in m whenever $m \geq n \log \log \ldots \log n$ for a constant number of 'log's. Hence it is at least as good a bound as $O(m + n \log n)$, for any setting of the parameters m, n.

3 Preliminaries

Definition 1. *A* deterministic graphical game (DGG) *is a directed graph with vertices partitioned into sets of non-terminals V_{Min} and V_{Max}, which are game positions where player* Min *and* Max*, respectively, chooses the next move (arc), and terminals T, where the game ends and* Min *pays* Max *the amount specified by $p : T \to \mathbf{R}$. There is a specified starting position v_0.*

For simplicity, we will assume that terminals have distinct payoffs, i.e., that p is injective. We can easily ensure this by artificially distinguishing terminals with equal payoffs in some arbitrary (but consistent) fashion. We will also assume that $m \geq n$, since terminals without incoming arcs are irrelevant.

Definition 2. *We denote by* $\text{Val}_G(v)$ *the value of position v in the game G.*

For now, we will focus on computing values of positions. We shall later see how to construct optimal *strategies* from these values. The definitions of a *strong* and a *weak* solution are as stated in Section 1.1.

The retrograde analysis algorithm described in Section 1.2 can be extended to DGGs by adding the following selection criterion: Among all candidate arcs (u, v), we choose one that maximizes the absolute value of the label of v. The algorithm thus obtained will be referred to as *generalized retrograde analysis*.

To extend our correctness proof, we must simply add to part 3 of the invariant: "Also, for each red arc (u, v) with u belonging to the player, the strategy guarantees the player a payoff at least as good as the label of v." For chess-like games, this property follows from part 2 of the invariant.

For an efficient implementation, we can first order all terminals by payoff in a list. Then, starting from the most positive or most negative terminal in the list, we repeatedly analyze incoming arcs and propagate labels whenever possible. After this, we can remove the terminal from the list, and repeat the process. This gives the following.

Proposition 1. *Given a DGG and a permutation that orders its terminals, we can find a strong solution to the game in linear time.*

We shall refer to this as the *sorting method* for strongly solving DGGs: First sort the payoffs, and then apply Proposition 1.

Corollary 1. *A DGG with m arcs and n terminals can be strongly solved in $O(m + n \log n)$ time.*

We will be interested in reducing the number of terminals in a game. To that end, we introduce the notion of *merging* terminals.

Definition 3. *To* merge *a terminal s into another terminal t is to reconnect all incoming arcs of s to t and then remove s. Two terminals are* adjacent *if their payoffs have the same sign and no other terminal has a payoff in between.*

The following lemma states the intuitive fact that when we merge two adjacent terminals, the only non-terminals affected are those with the corresponding values, and they acquire the same value.

Lemma 1. *If G' is obtained from the DGG G by merging a terminal s into an adjacent terminal t, then any optimal strategy in G is also optimal in G', and for each non-terminal v, we have*

$$\mathrm{Val}_{G'}(v) = \begin{cases} \mathrm{Val}_G(v) & \text{if } \mathrm{Val}_G(v) \neq p(s), \\ p(t) & \text{if } \mathrm{Val}_G(v) = p(s). \end{cases} \tag{1}$$

Proof. Assume that $0 < p(s) < p(t)$ (the other cases are similar), and let σ and τ be optimal strategies in G for Max and Min, respectively. It is easy to verify the following for any choice of non-terminal starting position v.

- If $\mathrm{Val}_G(v) \geq p(t)$, then σ ensures $\mathrm{Val}_G(v)$ or more in both G and G', and τ ensures $\mathrm{Val}_G(v)$ or less in both G and G'.
- If $\mathrm{Val}_G(v) = p(s)$, then σ ensures $p(s)$ or more in G, hence $p(t)$ or more in G', and τ ensures $p(s)$ or less in G, hence $p(t)$ or less in G'.

Figure 3: Coarsening by merging $\{-4, -1\}$ and $\{2, 3, 5\}$.

- If $\mathrm{Val}_G(v) < p(s)$, then σ ensures $\mathrm{Val}_G(v)$ or more in both G and G', and τ ensures $\mathrm{Val}_G(v)$ or less in both G and G'.

□

By repeatedly merging adjacent terminals, we can merge a set of terminals into one. We call this process *coarsening*. By Lemma 1, which describes the relation between the original game and the resulting coarse game, the order in which the terminals in a set are merged will be irrelevant. Figure 3 shows an example of this.

Lemma 2. *The signs of the values of all positions in a given DGG can be determined in linear time.*

Proof. Merge all terminals with positive payoffs into one, do likewise for those with negative payoffs, and then solve the resulting coarse game by generalized retrograde analysis. By Lemma 1, the values in the coarse game will have the same sign as the corresponding values in the original game. □

Clearly, arcs between positions with different values cannot be part of a strong solution to a game. From this, the following lemma is immediate.

Lemma 3. *In a DGG, removing an arc between two positions with different values does not affect the value of any position.*

Lemma 2, Lemma 3, and symmetry together allow us to restrict our attention to games where all positions have positive values, as will be done in subsequent sections.

Proposition 2. *Given the value of the starting position of a DGG, a weak solution can be found in linear time. If the values of* all *positions are known, a strong solution can be found in linear time.*

Proof. In the first case, let z be the value of the starting position v_0. We partition the payoffs in at most five intervals: $(-\infty, \min(z, 0))$, $\{\min(z, 0)\}$, $(\min(z, 0), \max(z, 0))$, $\{\max(z, 0)\}$ and $(\max(z, 0), \infty)$. We merge all terminals in each of the intervals, obtaining a game with at most five terminals. A strong solution for the resulting coarse game is found in linear time by generalized retrograde analysis. The pair of strategies obtained is then a weak solution to the original game, by Lemma 1.

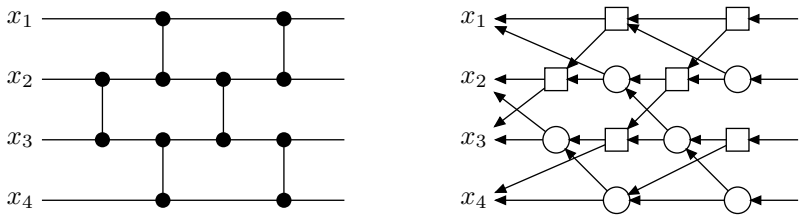


Figure 4: Implementing a sorting network by a deterministic graphical game.

In the second case, by Lemma 3, we can first discard all arcs between positions of different values. This disintegrates the game into smaller games where all positions have the same value. We find a strong solution to each of these games in linear time using generalized retrograde analysis. The union of these solutions is a strong solution to the original game, by Lemma 3. □

4 Solving Deterministic Graphical Games

4.1 Strongly

For solving DGGs in the strong sense, we currently know no asymptotically faster method than completely sorting the payoffs. Also, the number of *comparisons* this method performs is, when we consider bounds *only depending on the number of terminals* n, optimal. Any sorting network [22] can be implemented by an acyclic DGG, by simulating each comparator by a Max-vertex and a Min-vertex. Figure 4 shows an example of this. Thus, we have the following tight bound.

Proposition 3. *Strongly solving a DGG with n terminals requires $\Theta(n \log n)$ comparisons in the worst case.*

Implementing the asymptotically optimal AKS-network [1] results in a game with $\Theta(n \log n)$ positions and arcs. Thus, it is still consistent with our current knowledge that a DGG can be strongly solved using $O(m)$ comparisons.

4.2 Weakly

The algorithms we propose for weakly solving DGGs all combine coarsening of the set of payoffs with generalized retrograde analysis. By splitting the work between these two operations in different ways, we get different time/comparison trade-offs. At one extreme is the sorting method; at the other is the following algorithm due to Mike Paterson (personal communication). As noted in Section 3, we can assume that all positions have positive values.

Algorithm. Given a DGG G with m arcs, n terminals, and starting position v_0, do the following for $i = 0, 1, 2, \ldots$

1. Partition the current set of n_i terminals around their median payoff.

2. Solve the coarse game obtained by merging the terminals in each half.
3. Remove all positions that do not have values in the half containing $\mathrm{Val}_G(v_0)$.
4. Undo the merging of step 2 for the half containing $\mathrm{Val}_G(v_0)$.

When only one terminal remains, return its payoff as the value of v_0.

Analysis. By Lemma 1, solving the coarse game correctly identifies which half of the terminals contains the value of v_0, and by Lemma 3, removing the positions in step 3 does not affect the remaining positions.

Step 1 can be performed in $O(n_i)$ time (and comparisons) using the algorithm of Blum et al. [7]. Step 2 takes $O(m)$ time using generalized retrograde analysis. Since n_i is halved in each step, the number of iterations is $O(\log n)$, and the total running time is $O(m \log n)$—worse than the sorting method. However, the number of comparisons is $O(n + n/2 + n/4 + \ldots) = O(n)$, which is optimal.

4.2.1 $O(n)$ Comparisons and $O(m \log\log n)$ Time

To improve the running time of Paterson's algorithm, we stop and solve the remaining game using the sorting method as soon as $n_i \log n_i \leq n$. This final step takes $O(m)$ time and $O(n)$ comparisons.

The number of iterations becomes $O(\log n - \log f(n))$, where $f(n)$ is the inverse of $n \mapsto n \log n$, and since this equals $O(\log\log n)$ we have the following.

Theorem 1. *A DGG with m arcs and n terminals can be weakly solved in $O(m \log\log n)$ time and $O(n)$ comparisons.*

4.2.2 Almost-Linear Time

We can balance the partitioning and generalized retrograde analysis to achieve an almost-linear running time, by a technique similar to the one used by Gabow and Tarjan [12] and later generalized by Punnen [29]. Again, we assume that all positions have positive values.

Algorithm. Given a DGG G with m arcs, n terminals, and starting position v_0, do the following for $i = 0, 1, 2, \ldots$

1. Partition the current set of n_i terminals into groups of size at most $n_i 2^{-\frac{m}{n_i}}$.
2. Solve the coarse game obtained by merging the terminals in each group.
3. Remove all positions having values outside the group of $\mathrm{Val}_G(v_0)$.
4. Undo the merging of step 2 for the group of v_0.

When $n_i 2^{-\frac{m}{n_i}} < 1$, stop and solve the remaining game by the sorting method.

Analysis. All steps can be performed in $O(m)$ time. In particular, for the first step we can do a "partial perfect quicksort", where we always partition around the median and stop at level $\lceil m/n_i \rceil + 1$.

To bound the number of iterations, we note that n_i satisfies the recurrence

$$n_{i+1} \leq n_i 2^{-\frac{m}{n_i}}, \tag{2}$$

which by induction gives

$$n_i \leq \frac{n}{\underbrace{b^{b^{\cdot^{\cdot^{\cdot^{b}}}}}}_{i}} \tag{3}$$

where $b = 2^{m/n}$. Thus, the number of iterations is $O(\log_b^* n)$, where $\log_b^*$ denotes the number of times we need to apply the base b logarithm function to get below 1. This is easily seen to be the same as $O(1 + \log^* m - \log^* \frac{m}{n})$. We have now established the following.

Theorem 2. *A DGG with m arcs and n terminals can be weakly solved in $O(m + m(\log^* m - \log^* \frac{m}{n}))$ time.*

When $m = \Omega(n \log^{(k)} n)$ for some constant k, this bound is $O(m)$.

Acknowledgements. We are indebted to Mike Paterson for his contributions to this work. We also thank Uri Zwick for helpful discussions.

References

[1] M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. In *Proceedings of the 15th Annual ACM Symposium on the Theory of Computing*, pages 1–9, 1983.

[2] D. Andersson and P. Miltersen. The complexity of solving stochastic games on graphs. In preparation.

[3] V. Baston and F. Bostock. On Washburn's deterministic graphical games. In *Differential Games — Developments in Modelling and Computation.* Springer, 1991.

[4] R. Bellman. On the application of dynamic programming to the determination of optimal play in chess and checkers. *Proceedings of the National Academy of Sciences of the United States of America*, 53(2):244–246, 1965.

[5] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for your Mathematical Plays.* Academic Press, 1982.

[6] H. Björklund and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155(2):210–229, 2007.

[7] M. Blum, R. Floyd, V. Pratt, R. Rivest, and R. Tarjan. Linear time bounds for median computations. In *Proceedings of the 4th Annual ACM Symposium on the Theory of Computing*, pages 119–124, 1972.

[8] K. Chatterjee and T. A. Henzinger. Reduction of stochastic parity to stochastic mean-payoff games. *Inf. Process. Lett.*, 106(1):1–7, 2008.

[9] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[10] H. Everett. Recursive games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games Vol. III*, volume 39 of *Annals of Mathematical Studies*. Princeton University Press, 1957.

[11] C. Ewerhart. Backward induction and the game-theoretic analysis of chess. *Games and Economic Behavior*, 39:206–214, 2002.

[12] H. Gabow and R. Tarjan. Algorithms for two bottleneck optimization problems. *J. Algorithms*, 9:411–417, 1988.

[13] D. Gillette. Stochastic games with zero stop probabilities. In M. Dresher, A. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games III*, volume 39 of *Annals of Mathematics Studies*, pages 179–187. Princeton University Press, 1957.

[14] E. Grädel and W. Thomas. *Automata, Logics, and Infinite Games: A Guide to Current Research*. Springer, 2002.

[15] N. Halman. Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems. *Algorithmica*, 49(1):37–50, 2007.

[16] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 2:100–107, 1968.

[17] N. Immerman. *Descriptive complexity*. Springer, 1998.

[18] L. Kalmár. Zur Theorie der abstrakten Spiele. *Acta Sci. Math. Szeged*, 4:65–85, 1928.

[19] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 253–260, 2001.

[20] A. Kishimoto and M. Müller. A general solution to the graph history interaction problem. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI 2004)*, pages 644–649, 2004.

[21] D. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley, 1968.

[22] D. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 3rd edition, 1997.

[23] D. Knuth and R. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6:293–326, 1975.

[24] D. König. Über eine Schlussweise aus dem Endlichen ins Unendliche. *Acta Sci. Math. Szeged*, 3:121–130, 1927.

[25] T. M. Liggett and S. A. Lippman. Stochastic games with perfect information and time average payoff. *SIAM Review*, 11(4):604–607, 1969.

[26] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117(1):151–155, 1995.

[27] T. Niblett and A. Roycroft. How the GBR class 0103 data base was created. *EG*, 56, 1979.

[28] A. Palay. *Searching with Probabilities*. PhD thesis, Carnegie Mellon University, 1983.

[29] A. Punnen. A fast algorithm for a class of bottleneck problems. *Computing*, 56:397–401, 1996.

[30] J. Romein and H. Bal. Solving the game of awari using parallel retrograde analysis. *IEEE Computer*, 36(10):26–33, 2003.

[31] J. Schaeffer, N. Burch, Y. Bjornsson, A. Kishimoto, M. Muller, R. Lake, P. Lu, and S. Sutphen. Checkers is solved. *Science*, 2007.

[32] U. Schwalbe and P. Walker. Zermelo and the early history of game theory. *Games and Economic Behavior*, 34:123–137, 2001.

[33] L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences, U.S.A.*, 39:1095–1100, 1953.

[34] L. Stiller. Multilinear algebra and chess endgames. In *Games of No Chance*. MSRI Publications, 1996.

[35] T. Ströhlein. *Untersuchungen über kombinatorische Spiele*. PhD thesis, Technischen Hochschule München, 1970.

[36] K. Thompson. Retrograde analysis of certain endgames. *Journal of the International Computer Chess Association*, 9(3):131–139, 1986.

[37] E. van Damme. *Stability and Perfection of Nash Equlibria*. Springer-Verlag, 2nd edition, 1991.

[38] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press, Princeton, 1953. 3rd edition.

[39] A. Washburn. Deterministic graphical games. *Journal of Mathematical Analysis and Applications*, 153:84–96, 1990.

[40] E. Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proceedings of the Fifth International Congress of Mathematicians*, pages 501–504. Cambridge University Press, 1913.

[41] U. Zwick and M. S. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1–2):343–359, 1996.

Paper 2

The Complexity of Solving Stochastic Games on Graphs

Daniel Andersson, Peter Bro Miltersen

Abstract. We consider well-known families of two-player zero-sum perfect-information stochastic games played on finite directed graphs. Generalizing and unifying results of Liggett and Lippman, Zwick and Paterson, and Chatterjee and Henzinger, we show that the following tasks are polynomial-time equivalent.

- Solving *stochastic parity games*,
- Solving *simple stochastic games*,
- Solving *stochastic terminal-payoff games* with payoffs and probabilities given in unary,
- Solving stochastic terminal-payoff games with payoffs and probabilities given in binary,
- Solving *stochastic mean-payoff games* with rewards and probabilities given in unary,
- Solving stochastic mean-payoff games with rewards and probabilities given in binary,
- Solving *stochastic discounted-payoff games* with discount factor, rewards and probabilities given in binary.

It is unknown whether these tasks can be performed in polynomial time. In the above list, "solving" may mean either *quantitatively* solving a game (computing the values of its positions) or *strategically* solving a game (computing an optimal strategy for each player). In particular, these two tasks are polynomial-time equivalent for all the games listed above. We also consider a more refined notion of equivalence between quantitatively and strategically solving a game. We exhibit a linear time algorithm that given a simple stochastic game or a terminal-payoff game *and* the values of all positions of that game, computes a pair of optimal strategies. Consequently, for any restriction one may put on the simple stochastic game model, quantitatively solving is polynomial-time equivalent to strategically solving the resulting class of games.

1 Introduction

We consider some well-known families of two-player zero-sum perfect-information stochastic games played on finite directed graphs.

- *Simple stochastic games* were introduced to the algorithms and complexity community by Condon [6], who was motivated by the study of randomized Turing machine models. A simple stochastic game is given by a finite directed graph $G = (V, E)$, with the set of vertices V also called *positions* and the set of arcs E also called *actions*. There is a partition of the positions into V_1 (positions belonging to Player 1), V_2 (positions belonging to Player 2), V_R (coin-toss positions), and a special terminal position **1**. Positions of V_R have exactly two outgoing arcs, the terminal position **1** has none, while all positions in V_1, V_2 have at least one outgoing arc. Between moves, a pebble is resting at some vertex u. If u belongs to a player, this player should strategically pick an outgoing arc from u and move the pebble along this edge to another vertex. If u is a vertex in V_R, nature picks an outgoing arc from u uniformly at random and moves the pebble along this arc. The objective of the game for Player 1 is to reach **1** and should play so as to maximize his probability of doing so. The objective for Player 2 is to prevent Player 1 from reaching **1**.

- *Stochastic terminal-payoff games* is the natural generalization of simple stochastic games where we allow

  1. each vertex in V_R to have more than two outgoing arcs and an arbitrary rational valued probability distribution on these,
  2. several terminals with different payoffs, positive or negative.

  In a stochastic terminal-payoff game, the *outcome* of the game is the payoff of the terminal reached, with the outcome being 0 if play is infinite. Generalizing the objectives of a simple stochastic game in the natural way, the objective for Player 1 is to maximize the expected outcome while the objective for Player 2 is to minimize it. Such generalized simple stochastic games have sometimes also been called simple stochastic games in the literature (e.g., [10]), but we shall refer to them as stochastic terminal-payoff games in this paper.

- *Stochastic parity games* were introduced by Chatterjee, Jurdziński, and Henzinger at SODA'04 [4] and further studied in [3, 2]. They are a natural generalization of the non-stochastic parity games of McNaughton [16], the latter having central importance in the computer-aided verification community, as solving them is equivalent to model checking the μ-calculus [7]. As for the case of simple stochastic games, a stochastic parity game is given by a directed graph $G = (V, E)$ with a partition of the vertices into V_1 (vertices belonging to Player 1), V_2 (vertices belonging to Player 2), and V_R (random vertices). Vertices of V_R have exactly two outgoing arcs, while all vertices in V_1, V_2 have at least one outgoing arc. Also,

each vertex is assigned an integral *priority*. Between moves, a pebble is resting at some vertex u. If u belongs to a player, this player should strategically pick an outgoing arc from u and move the pebble along this edge to another vertex. If u is a vertex in V_R, nature picks an outgoing arc from u uniformly at random and moves the pebble along this arc. The play continues forever. If the highest priority that appears infinitely often during play is odd, Player 1 wins the game; if it is even, Player 2 wins the game.

- *Stochastic mean-payoff games* and *stochastic discounted-payoff games* were first studied in the game theory community by Gillette [9] as the perfect information special case of the stochastic games of Shapley [17]. A stochastic mean-payoff or discounted-payoff game G is given by a finite set of positions V, partitioned into V_1 (positions belonging to Player 1) and V_2 (positions belonging to Player 2). To each position u is associated a finite set of possible actions. To each such action is associated a real-valued *reward* and a probability distribution on positions. At any point in time of play, the game is in a particular position i. The player to move chooses an action strategically and the corresponding reward is paid by Player 2 to Player 1. Then, nature chooses the next position at random according to the probability distribution associated with the action. The play continues forever and the sum of rewards may therefore be unbounded. Nevertheless, one can associate a finite payoff to the players in spite of this, in more ways than one (so G is not just one game, but really a family of games): For a stochastic discounted-payoff game, we fix a *discount factor* $\beta \in (0, 1)$ and define the outcome of the play (the payoff to Player 1) to be
$$\sum_{i=0}^{\infty} \beta^i r_i$$
where r_i is the reward incurred at stage i of the game. We shall denote the resulting game G_β. For a stochastic mean-payoff game we define the outcome of the play (the payoff to Player 1) to be the *limiting average* payoff
$$\liminf_{n\to\infty} (\sum_{i=0}^{n} r_i)/(n+1).$$
We shall denote the resulting game G_1. A natural restriction of stochastic mean-payoff games is to *deterministic* transitions (i.e., all probability distributions put all probability mass on one position). This class of games has been studied in the computer science literature under the names of cyclic games [11] and mean-payoff games [18]. We shall refer to them as deterministic mean-payoff games in this paper.

A *strategy* for a game is a (possibly randomized) procedure for selecting which arc or action to take, given the history of the play so far. A *pure positional strategy* is the very special case of this where the choice is deterministic and only depends on the current position, i.e., a pure positional strategy is simply a map

from positions to actions. If Player 1 plays using strategy x and Player 2 plays using strategy y, and the play starts in position i, a random play $P(x, y, i)$ of the game is induced. We let $u^i(x, y)$ denote the expected outcome of this play (for stochastic terminal-payoff, discounted-payoff, and mean-payoff games) or the winning probability of Player 1 (for simple stochastic games and stochastic parity games). A strategy x^* for Player 1 is said to be *optimal* if for all positions i it holds that,

$$\inf_{y \in S_2} u^i(x^*, y) \geq \sup_{x \in S_1} \inf_{y \in S_2} u^i(x, y) \tag{1}$$

where S_1 (S_2) is the set of strategies for Player 1 (Player 2). Similarly, a strategy y^* for Player 2 is said to be optimal if

$$\sup_{x \in S_1} u^i(x, y^*) \leq \inf_{y \in S_2} \sup_{x \in S_1} u^i(x, y). \tag{2}$$

For all games described here, the references above (a proof of Liggett and Lippman [14] fixes a bug of a proof of Gillette [9] for the mean-payoff case) show that

- Both players have pure positional optimal strategies x^*, y^*.
- For such optimal x^*, y^* and for all positions i,

$$\inf_{y \in S_2} u^i(x^*, y) = \sup_{x \in S_1} u^i(x, y^*).$$

  This number is called the *value* of position i. We shall denote it $\mathrm{val}(i)$.

These facts imply that when testing whether conditions (1) and (2) hold, it is enough to take the infima and suprema over the finite set of pure positional strategies of the players.

In this paper, we consider *solving* games. By solving a game G, we may refer to two distinct tasks.

- *Quantitatively solving* G is the task of computing the values of all positions of the game, given an explicit representation of G.
- *Strategically solving* G is the task of computing a pair of optimal positional strategies for the game, given an explicit representation of G.

To be able to explicitly represent the games, we assume that the discount factor, rewards and probabilities are rational numbers and given as fractions in binary or unary. The notion of "quantitatively solving" is standard terminology in the literature (e.g., [4]), while the notion of "strategically solving" is not standard. We believe the distinction is natural, in particular since recent work of Hansen *et al.* [12] shows that for some classes of games more general than the ones considered in this paper, the two notions are not equivalent. Still, it is relatively easy to see that for all the games considered here, once the game has been strategically solved, we may easily solve it quantitatively.

With so many distinct computational problems under consideration, we shall for this paper introduce some convenient notation for them. We will use

superscripts q/s to distinguish quantitative/strategic solutions and subscripts b/u to distinguish binary/unary input encoding (when applicable). For instance, $\textsc{Mean}^{\text{s}}_{\text{b}}$ is the problem of solving a stochastic mean-payoff game strategically with probabilities and rewards given in binary, and $\textsc{Simple}^{\text{q}}$ is the problem of solving a simple stochastic game quantitatively. We use "$\preceq$" to express polynomial-time (Turing) reducibility.

Solving the games above in polynomial time are all celebrated open problems (e.g., [6, 4]). Also, some polynomial time reductions between these challenging tasks were known. Some classes of games are obviously special cases of others (e.g., simple stochastic games and stochastic terminal-payoff games), leading to trivial reductions, but more intricate reductions were also known. In particular, a recent paper by Chatterjee and Henzinger [2] shows that solving stochastic parity games reduces to solving stochastic mean-payoff games. Earlier, Zwick and Paterson [18] showed that solving *deterministic* mean-payoff games reduces to solving simple stochastic games. However, in spite of these reductions and the fact that similar kinds of (non-polynomial time) algorithms, such as value iteration and strategy iteration are used to actually solve all of these games in practice, it does not seem that it was believed or even suggested that the tasks of solving these different classes of games were all polynomial-time equivalent. Our first main result is that they are, unifying and generalizing the previous reductions (and also using them in a crucial way):

Theorem 1. *The following tasks are polynomial-time (Turing) equivalent.*

- *Solving stochastic parity games,*
- *Solving simple stochastic games,*
- *Solving stochastic terminal-payoff games with payoffs and probabilities given in unary,*
- *Solving stochastic terminal-payoff games with payoffs and probabilities given in binary,*
- *Solving stochastic mean-payoff games with rewards and probabilities given in unary,*
- *Solving stochastic mean-payoff games with rewards and probabilities given in binary,*
- *Solving stochastic discounted-payoff games with discount factor, rewards and probabilities given in binary.*

Solving here may mean either "quantitatively" or "strategically". In particular, the two tasks are polynomial-time equivalent for all these classes of games.

Note that the equivalence between solving games with unary input encoding and solving games with binary input encoding means that there are pseudopolynomial-time algorithms for solving these games if and only there are polynomial-time algorithms. Note also that a "missing bullet" in the theorem

is solving stochastic discounted-payoff games given in unary representation. It is in fact known that this can be done in polynomial time (even if only the discount factor is given in unary while rewards and probabilities are given in binary), see Littman [15, Theorem 3.4].

One obvious interpretation of our result is that Condon "saw right" when she singled out simple stochastic games as a representative model. In fact, our result implies that if simple stochastic games can be solved in polynomial time, then essentially all important classes of two-player zero-sum perfect-information stochastic games considered in the literature can also be solved in polynomial time.

Our second main result takes a closer look at the equivalence between quantitatively solving the games we consider and strategically solving them. Even though Theorem 1 shows these two tasks to be polynomial-time equivalent, the reductions from strategically solving games to quantitatively solving games in general changes the game under consideration, i.e., strategically solving one game reduces to quantitatively solving another. In other words, Theorem 1 does *not* mean *a priori* that once a game has been quantitatively solved, we can easily solve it strategically. However, for the case of discounted-payoff games, we trivially can: An optimal action can be determined "locally" by comparing sums of local rewards and values. Our second result shows that we (less trivially) can do such "strategy recovery" also for the case of terminal-payoff games (and therefore also for simple stochastic games).

Theorem 2. *Given a stochastic terminal-payoff game and the values of all its vertices, a pair of optimal pure positional strategies can be computed in linear time.*

Theorem 2 means that algorithms that efficiently and quantitatively solve simple stochastic games satisfying certain conditions (such as the algorithm of Gimbert and Horn [10], which is efficient when the number of coin-toss vertices is small) can also be used to strategically solve those games, in essentially the same time bound. We leave as an open problem whether a similar algorithm (even a polynomial-time one) can be obtained for stochastic mean-payoff games.

2 Proof of Theorem 1

Figure 1 shows a minimal set of reductions needed to establish all equivalences. We first enumerate a number of trivial and known reductions and afterwards fill in the remaining "gaps".

For all games considered here, it is well-known that quantitatively solving them reduces to strategically solving them: Once the strategies have been fixed, a complete analysis of the resulting finite-state random process can be obtained using linear algebra and the theory of Markov chains [13]. Also, for the case of stochastic discounted-payoff games, the converse reduction is also easy: A strategy is optimal if and only it in each position chooses an action that maximizes the sum of the reward obtained by this action and the discounted expected value of the next position. Thus, $\textsc{Discounted}^{\mathrm{s}}_{\mathrm{b}} \preceq \textsc{Discounted}^{\mathrm{q}}_{\mathrm{b}}$.

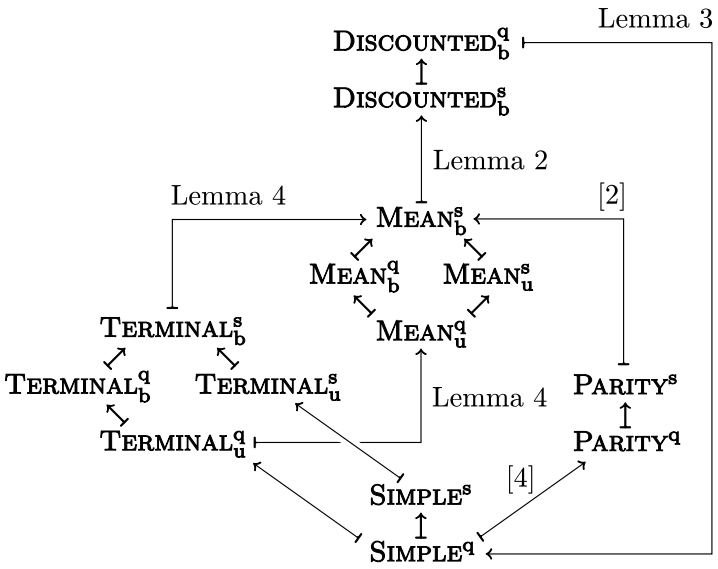


Figure 1: Reductions used in the proof of Theorem 1.

Of course, each problem with unary input encoding trivially reduces to the corresponding binary version. Also, it is obvious that stochastic terminal-payoff games generalize simple stochastic games, and the only numbers that appear are 0, 1 and $\frac{1}{2}$, so $\textsc{Simple}^{\mathrm{q}} \preceq \textsc{Terminal}^{\mathrm{q}}_{\mathrm{u}}$ and $\textsc{Simple}^{\mathrm{s}} \preceq \textsc{Terminal}^{\mathrm{s}}_{\mathrm{u}}$.

Quantitatively solving simple stochastic games easily reduces to quantitatively solving stochastic parity games, as was noted in the original work on stochastic parity games [4]. Also, Chatterjee and Henzinger [2] show that strategically solving stochastic parity games reduces to strategically solving stochastic mean-payoff games.

Thus, to "complete the picture" and establish all equivalences, we only have to show:

- $\textsc{Mean}^{\mathrm{s}}_{\mathrm{b}} \preceq \textsc{Discounted}^{\mathrm{s}}_{\mathrm{b}}$,
- $\textsc{Discounted}^{\mathrm{q}}_{\mathrm{b}} \preceq \textsc{Simple}^{\mathrm{q}}$,
- $\textsc{Terminal}^{\mathrm{s}}_{\mathrm{b}} \preceq \textsc{Mean}^{\mathrm{s}}_{\mathrm{b}}$ and $\textsc{Terminal}^{\mathrm{q}}_{\mathrm{u}} \preceq \textsc{Mean}^{\mathrm{q}}_{\mathrm{u}}$.

These reductions are provided by the following lemmas.

Lemma 1. *Let G be a stochastic discounted-payoff/mean-payoff game with n positions and all transition probabilities and rewards being fractions with integral numerators and denominators, all of absolute value at most M. Let $\beta^* = 1 - ((n!)^2 2^{2n+3} M^{2n^2})^{-1}$ and let $\beta \in [\beta^*, 1)$. Then, any optimal pure positional strategy (for either player) in the discounted-payoff game G_β is also an optimal strategy in the mean-payoff game G_1.*

Proof. The fact that *some* β^* with the desired property exists is explicit in the proof of Theorem 1 of Liggett and Lippman [14]. We assume familiarity with

that proof in the following. Here, we derive a concrete value for β^*. From the proof of Liggett and Lippman, we have that for x^* to be an optimal pure positional strategy (for Player 1) in G_1, it is sufficient to be an optimal pure positional strategy in G_β for all values of β sufficiently close to 1, i.e., to satisfy the inequalities

$$\min_{y \in S_2'} u_\beta^i(x^*, y) \geq \max_{x \in S_1'} \min_{y \in S_2'} u_\beta^i(x, y)$$

for all positions i and for all values of β sufficiently close to 1, where S_1' (S_2') is the set of pure positional, strategies for Player 1 (2) and u_β^i is the expected payoff when game starts in position i and the discount factor is β. Similarly, for y^* to be an optimal pure positional strategy (for Player 2) in G_1, it is sufficient to be an optimal pure positional strategy in G_β for all values of β sufficiently close to 1, i.e., to satisfy the inequalities

$$\max_{x \in S_1'} u_\beta^i(x, y^*) \leq \min_{y \in S_2'} \max_{x \in S_1'} u_\beta^i(x, y).$$

So, we can prove the lemma by showing that for all positions i and *all* pure positional strategies x, y, x', y', the sign of $u_\beta^i(x, y) - u_\beta^i(x', y')$ is the same for all $\beta \geq \beta^*$. For fixed strategies x, y we have that $v_i = u_\beta^i(x, y)$ is the expected total reward in a *discounted Markov process* and is therefore given by the formula (see [13])

$$v = (I - \beta Q)^{-1} r, \tag{3}$$

where v is the vector of $u_\beta(x, y)$ values, one for each position, Q is the matrix of transition probabilities and r is the vector of rewards (note that for *fixed* positional strategies x, y, rewards can be assigned to positions in the natural way). Let $\gamma = 1 - \beta$. Then, (3) is a system of linear equations in the unknowns v, where each coefficient is of the form $a_{ij}\gamma + b_{ij}$ where a_{ij}, b_{ij} are rational numbers with numerators with absolute value bounded by $2M$ and with denominators with absolute value bounded by M. By multiplying the equations with all denominators, we can in fact assume that a_{ij}, b_{ij} are integers of absolute value less than $2M^n$. Solving the equations using Cramer's rule, we may write an entry of v as a quotient between determinants of $n \times n$ matrices containing terms of the form $a_{ij}\gamma + b_{ij}$. The determinant of such a matrix is a polynomial in γ of degree n with the coefficient of each term being of absolute value at most $n!(2M^n)^n = n!2^n M^{n^2}$. We denote these two polynomials p_1, p_2. Arguing similarly about $u_\beta(x', y')$ and deriving corresponding polynomials p_3, p_4, we have that $u_\beta^i(x, y) - u_\beta^i(x', y') \geq 0$ is equivalent to $p_1(\gamma)/p_2(\gamma) - p_3(\gamma)/p_4(\gamma) \geq 0$, i.e., $p_1(\gamma)p_4(\gamma) - p_3(\gamma)p_2(\gamma) \geq 0$. Letting $q(\gamma) = p_1(\gamma)p_4(\gamma) - p_3(\gamma)p_2(\gamma)$, we have that q is a polynomial in γ, with integer coefficients, all of absolute value at most $R = 2(n!)^2 2^{2n} M^{2n^2}$. Since $1 - \beta^* < 1/(2R)$, the sign of $q(\gamma)$ is the same for all $\gamma \leq 1 - \beta^*$, i.e., for all $\beta \geq \beta^*$. This completes the proof. □

Lemma 2. $\textsc{Mean}_{\mathrm{b}}^{\mathrm{s}} \preceq \textsc{Discounted}_{\mathrm{b}}^{\mathrm{s}}$.

Proof. This follows immediately from Lemma 1 by observing that the binary representation of the number $\beta^* = 1 - ((n!)^2 2^{2n+3} M^{2n^2})^{-1}$ has length polynomial in the size of the representation of the given game. □

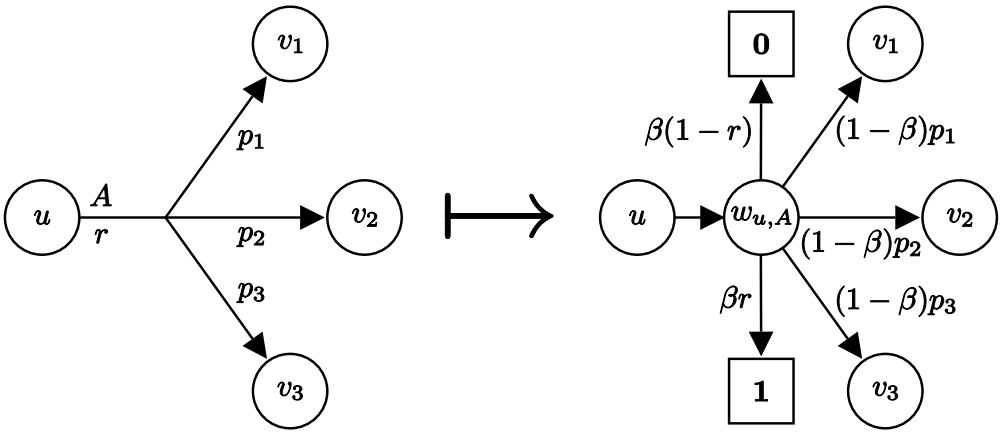


Figure 2: Reducing discounted-payoff games to terminal-payoff games

Lemma 3. $\textsc{Discounted}^{\text{q}}_{\text{b}} \preceq \textsc{Simple}^{\text{q}}$.

Proof. Zwick and Paterson [18] considered solving *deterministic* discounted-payoff games, i.e., games where the action taken deterministically determines the transition taken and reduced these to solving simple stochastic games. It is natural to try to generalize their reduction so that it also works for stochastic discounted-payoff games. We find that such reduction indeed works, even though the correctness proof of Zwick and Paterson has to be modified slightly compared to their proof. The details follow.

The redution proceeds in two steps: First we reduce to stochastic terminal-payoff games with 0/1 payoffs, and then to simple stochastic games.

We are given as input a stochastic discounted-payoff game G with discount factor β and must first produce a stochastic terminal-payoff game G' whose values can be used to construct the values for the stochastic discounted-payoff game G_β. First, we affinely scale and translate all rewards of G so that they are in the interval $[0, 1]$. This does not influence the optimal strategies, and all values are transformed accordingly. Vertices of G' include all positions of G (belonging to the same player in G' as in G), and, in addition, a random vertex $w_{u,A}$ for each possible action A of each position u of G. We also add terminals $\mathbf{0}$ and $\mathbf{1}$. We construct the arcs of G' by adding, for each (position,action) pair (u, A) the "gadget" indicated in Figure 2. To be precise, if the action has reward r and leads to positions $v_1, v_2, \ldots, v_k$ with probability weights $p_1, p_2, \ldots, p_k$, we include in G' an arc from u to $w_{u,A}$, arcs from $w_{u,A}$ to $v_1, \ldots, v_k$ with probability weights $(1-\beta)p_1, \ldots, (1-\beta)p_k$, an arc from $w_{u,A}$ to $\mathbf{0}$ with probability weight $\beta(1-r)$ and finally an arc from $w_{u,A}$ to the terminal $\mathbf{1}$ with probability weight βr.

There is clearly a one-to-one correspondence between strategies in G and in G'. To see the correspondance between values, fix a strategy profile for the two players and consider play starting from some vertex. By construction, if the expected reward of the play in G is h, the probability that the play in G' reaches $\mathbf{1}$ is exactly βh.

The second step is identical to that of Zwick and Paterson [18]. They describe how arbitrary rational probability distributions can be implemented using a polynomial number of coin-toss vertices. Thus, we can transform G' into an equivalent simple stochastic game. □

Lemma 4. $\textsc{Terminal}^{\mathrm{s}}_{\mathrm{b}} \preceq \textsc{Mean}^{\mathrm{s}}_{\mathrm{b}}$ *and* $\textsc{Terminal}^{\mathrm{q}}_{\mathrm{u}} \preceq \textsc{Mean}^{\mathrm{q}}_{\mathrm{u}}$.

Proof. We are given a simple stochastic game G and must construct a stochastic mean-payoff game G'. Positions of G' will coincide with vertices of G, with the positions of G' including the terminals. Positions u belonging to a player in G belongs to the same player in G'. For each outgoing arc of u, we add an action in G' with reward 0, and with a deterministic transition to the endpoint of the arc of G. Random vertices of G can be assigned to either player in G', but he will only be given a single "dummy choice": If the random vertex has arcs to v_1 and v_2, we add a single action in G' with reward 0 and transitions into v_1, v_2, both with probability weight 1/2. Each terminal can be assigned to either player in G', but again he will be given only a dummy choice: We add a single action with reward equal to the payoff of the terminal and with a transition back into the same terminal with probability weight 1.

There is clearly a one-to-one correspondence between pure positional strategies in G and strategies in G'. To see the correspondence between values, fix a strategy profile for the two players and consider play starting from some vertex. By construction, if the probability of the play reaching some particular terminal in G is q, then the probability of the play reaching the corresponding self-loop in G' is also q. Thus, the expected reward is the same. □

3 Proof of Theorem 2

In this section, we consider a given stochastic terminal-payoff game. Our goal is an algorithm for computing optimal strategies, given the values of all vertices. To simplify the presentation, we will focus on strategies for Player 1. Because of symmetry, there is no loss of generality. In this section, "strategy" means "pure positional strategy".

Definition 1. *An arc (u, v) is called* safe *if* $\mathrm{val}(u) = \mathrm{val}(v)$. *A* safe strategy *is a strategy that only uses safe arcs.*

Definition 2. *A strategy x for Player 1 is called* stopping *if for any strategy y for Player 2 and any vertex v with positive value, there is a non-zero probability that the play $P(x, y, v)$ reaches a terminal.*

It is not hard to see that any optimal strategy for Player 1 must be safe and stopping, so these two conditions are necessary for optimality. We will now show that they are also sufficient.

Lemma 5. *If a strategy is safe and stopping, then it is also optimal.*

Proof. Let x be any safe and stopping strategy for Player 1, let y be an arbitrary strategy for Player 2, and let v_0 be an arbitrary vertex. Consider the play

$P(x, y, v_0)$. Denote by $q_i(v)$ the probability that after i steps, the play is at v. Since x is safe,

$$\forall i: \quad \mathrm{val}(v_0) \leq \sum_{v \in V} \mathrm{val}(v) q_i(v) \leq \sum_{v \in T \cup V^+} \mathrm{val}(v) q_i(v), \tag{4}$$

where T denotes the set of terminal vertices and V^+ denotes the set of non-terminal vertices with positive value. Since x is stopping,

$$\forall v \in V^+: \quad \lim_{i \to \infty} q_i(v) = 0. \tag{5}$$

Finally, note that

$$\begin{aligned} u^{v_0}(x, y) &= \sum_{t \in T} \mathrm{val}(t) \lim_{i \to \infty} q_i(t) \\ &= \sum_{v \in T \cup V^+} \mathrm{val}(v) \lim_{i \to \infty} q_i(v) && \text{(by (5))} \\ &= \lim_{i \to \infty} \sum_{v \in T \cup V^+} \mathrm{val}(v) q_i(v) \\ &\geq \mathrm{val}(v_0). && \text{(by (4))} \end{aligned}$$

Therefore, x is optimal. □

Using this characterization of optimality, strategy recovery can be reduced to strategically solving a simple stochastic game without random vertices.

Theorem 2. *Given a stochastic terminal-payoff game and the values of all its vertices, a pair of optimal pure positional strategies can be computed in linear time.*

Proof. Construct from the given game G a simple stochastic game G' as follows:

1. Merge all terminals into one.
2. Remove all outgoing non-safe arcs from the vertices of Player 1.
3. Transfer ownership of all random vertices to Player 1.

Compute an optimal strategy x' for Player 1 in G' using linear-time retrograde analysis [1]. Let x be the interpretation of x' as a strategy in G obtained by restricting x' to the vertices of Player 1 in G.

By construction, from any starting vertex v, Player 1 can ensure reaching the terminal in G' if and only if x ensures a non-zero probability of reaching a terminal in G.

Let v be any vertex with positive value in G. Player 1 has a safe strategy that ensures a non-zero probability of reaching a terminal from v, specifically, the optimal strategy. This implies that there is a corresponding strategy for Player 1 in G' that ensures reaching the terminal from v.

It follows that x is stopping, and it is safe by construction. Therefore, by Lemma 5, it is optimal.

□

4 Conclusions and Open Problems

Informally, we have shown that solving simple stochastic games is "complete" for a wide natural range of two-player zero-sum perfect-information games on graphs. However, in the logic and verification literature, even more general classes of winning conditions on infinite plays in finite graphs have been considered. Since we now know that simple stochastic games are more expressive than previously suggested, it would be interesting to fully characterize the class of such games for which solving simple stochastic game is complete. It is also interesting to ask whether solving some classes of *imperfect information* games reduces to solving simple stochastic games. It seems natural to restrict attention to cases where it is known that optimal positional strategies exists. This precludes general stochastic games (but see [5]). An interesting class of games generalizing stochastic mean-payoff games was considered by Filar [8]. Filar's games allow simultaneous moves by the two players. However, for any position, the probability distribution on the next position can depend on the action of one player only. Filar shows that his games are guaranteed to have optimal positional strategies. The optimal strategies are not necessarily pure, but the probabilities they assign to actions are guaranteed to be rational numbers if rewards and probabilities are rational numbers. So, we ask: Is solving Filar's games polynomial time equivalent to solving simple stochastic games?

Turning to strategy recovery, we mention again that we do not know whether the task of computing optimal strategies once values are known can be done in polynomial time for stochastic mean-payoff games. Recent work by Vladimir Gurvich (personal communication) indicates that the problem of computing optimal strategies in stochastic mean-payoff games is challenging even in the *ergodic* case where all positions have the same value. One may even hypothesise that strategy recovery for stochastic mean-payoff games is hard, in the (optimal) sense that it is polynomial time equivalent to strategically solving stochastic mean-payoff games. Establishing this would be most interesting.

Acknowledgements. We would like to thank Vladmir Gurvich for fruitful collaboration and Florian Horn for helpful comments.

References

[1] D. Andersson, K. A. Hansen, P. B. Miltersen, and T. B. Sørensen. Deterministic graphical games revisited. In *CiE '08: Proceedings of the 4th conference on Computability in Europe*, pages 1–10, Berlin, Heidelberg, 2008. Springer-Verlag.

[2] K. Chatterjee and T. A. Henzinger. Reduction of stochastic parity to stochastic mean-payoff games. *Inf. Process. Lett.*, 106(1):1–7, 2008.

[3] K. Chatterjee, M. Jurdziński, and T. Henzinger. Simple stochastic parity games. In *CSL: Computer Science Logic*, Lecture Notes in Computer Science 2803, pages 100–113. Springer, 2003.

[4] K. Chatterjee, M. Jurdziński, and T. A. Henzinger. Quantitative stochastic parity games. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 121–130, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[5] K. Chatterjee, R. Majumdar, and T. A. Henzinger. Stochastic limit-average games are in EXPTIME. International Journal of Game Theory, to appear.

[6] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[7] E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model checking for the mu-calculus and its fragments. *Theoretical Computer Science*, 258(1-2):491–522, May 2001.

[8] J. Filar. Ordered field property for stochastic games when the player who controls transitions changes from state to state. *Journal of Optimization Theory and Applications*, 34:503–513, 1981.

[9] D. Gillette. Stochastic games with zero stop probabilities. In M. Dresher, A. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games III*, volume 39 of *Annals of Mathematics Studies*, pages 179–187. Princeton University Press, 1957.

[10] H. Gimbert and F. Horn. Simple Stochastic Games with Few Random Vertices are Easy to Solve. In *Proceedings of the 11th International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'08*, volume 4962 of *Lecture Notes in Computer Science*, pages 5–19. Springer-Verlag, 2008.

[11] V. Gurvich, A. Karzanov, and L. Khachiyan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *USSR Computational Mathematics and Mathematical Physics*, 28:85–91, 1988.

[12] K. A. Hansen, M. Koucky, and P. B. Miltersen. Winning concurrent reachability games requires doubly-exponential patience. LICS'09, to appear., 2009.

[13] R. A. Howard. *Dynamic Programming and Markov Processes*. M.I.T. Press, 1960.

[14] T. M. Liggett and S. A. Lippman. Stochastic games with perfect information and time average payoff. *SIAM Review*, 11(4):604–607, 1969.

[15] M. L. Littman. *Algorithms for sequential decision making*. PhD thesis, Brown University, Department of Computer Science, 1996.

[16] R. McNaughton. Infinite games played on finite graphs. *An. Pure and Applied Logic*, 65:149–184, 1993.

[17] L. Shapley. Stochastic games. *Proc. Nat. Acad. Science*, 39:1095–1100, 1953.

[18] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1-2):343–359, 1996.

Paper 3

On Acyclicity of Games with Cycles

Daniel Andersson, Vladimir Gurvich, Thomas Dueholm Hansen

Abstract. We study restricted improvement cycles (ri-cycles) in finite positional n-person games with perfect information modeled by directed graphs (di-graphs) that may contain directed cycles (di-cycles). We obtain criteria of restricted improvement acyclicity (ri-acyclicity) in two cases: for $n = 2$ and for acyclic digraphs. We provide several examples that outline the limits of these criteria and show that, essentially, there are no other ri-acyclic cases. We also discuss connections between ri-acyclicity and some open problems related to Nash-solvability.

This is an extended version of our paper in *Proceedings of the Fifth International Conference on Algorithmic Aspects in Information and Management* (AAIM 2009), LNCS 5564.

1 Main Concepts and Results

1.1 Games in Normal Form

1.1.1 Game Forms, Strategies, and Utility Functions

Given a set of players $I = \{1, \dots, n\}$ and a set of strategies X_i for each $i \in I$, let $X = \prod_{i \in I} X_i$.

A vector $x = (x_i, i \in I) \in X$ is called a *strategy profile* or *situation*.

Furthermore, let A be a set of outcomes. A mapping $g : X \to A$ is called a *game form*. In this paper, we restrict ourselves to *finite* game forms, that is, we assume that sets I, A and X are finite.

Then, let $u : I \times A \to \mathbb{R}$ be a utility function. Standardly, the value $u(i, a)$ (or $u_i(a)$) is interpreted as the payoff to player $i \in I$ in case of the outcome $a \in A$. In figures, the notation $a <_i b$ means $u_i(a) < u_i(b)$.

Sometimes, it is convenient to exclude ties. Accordingly, u is called a *preference profile* if the mapping u_i is injective for each $i \in I$; in other words, u_i defines a complete order over A describing the preferences of player $i \in I$.

A pair (g, u) is called a *game in normal form*.

1.1.2 Improvement Cycles and Acyclicity

In a game (g, u), an *improvement cycle (im-cycle)* is defined as a sequence of k strategy profiles $\{x^1, \dots, x^k\} \subseteq X$ such that x^j and x^{j+1} coincide in all coordinates but one $i = i(j)$ and, moreover, $u_i(x^{j+1}) > u_i(x^j)$, that is, player i makes profit by substituting strategy x_i^{j+1} for x_i^j; this holds for all $j = 1, \dots, k$ and, standardly, we assume that $k + 1 = 1$.

A game (g, u) is called *im-acyclic* if it has no im-cycles. A game form g is called *im-acyclic* if for each u the corresponding game (g, u) is im-acyclic.

We call x^{j+1} an improvement with respect to x^j for player $i = i(j)$. We call it a *best reply* (BR) improvement if player i can get no strictly better result provided all other players keep their strategies. Correspondingly, we introduce the concepts of a BR im-cycle and BR im-acyclicity. Obviously, im-acyclicity implies BR im-acyclicity but not vice versa.

1.1.3 Nash Equilibria and Acyclicity

Given a game (g, u), a strategy profile $x \in X$ is called a *Nash equilibrium* (NE) if $u_i(x) \geq u_i(x')$ for each $i \in I$, whenever $x'_j = x_j$ for all $j \in I \setminus \{i\}$. In other words, x is a NE if no player can get a strictly better result by substituting a new strategy (x'_i for x_i) when all other players keep their old strategies. Conversely, if x is not a NE then there is a player who can improve his strategy. In particular, he can choose a best reply. Hence, a NE-free game (g, u) has a BR im-cycle. Let us remark that the last statement holds only for finite games, while the converse statement is not true at all.

A game (g, u) is called *Nash-solvable* if it has a NE. A game form g is called *Nash-solvable* if for each u the corresponding game (g, u) has a NE.

1.2 Positional Games with Perfect Information

1.2.1 Games in Positional Form

Let $G = (V, E)$ be a finite directed graph (di-graph) whose vertices $v \in V$ and directed edges $e \in E$ are called *positions* and *moves*, respectively. An edge $e = (v', v'')$ is a move from position v' to v''. Let $out(v)$ and $in(v)$ denote the sets of moves from and to v, respectively.

A position $v \in V$ is called *terminal* if $out(v) = \emptyset$, i.e., there are no moves from v. Let V_T denote the set of all terminals. Let us also fix a starting position $v_0 \in V \setminus V_T$. A directed path from v_0 to a terminal position is called a *finite play*.

Furthermore, let $D : V \setminus V_T \to I$ be a decision mapping, with $I = \{1, \ldots, n\}$ being the set of players. We say that the player $i = D(v) \in I$ makes a decision (move) in a position $v \in D^{-1}(i) = V_i$. Equivalently, D is defined by a partition $D : V = V_1 \cup \ldots \cup V_n \cup V_T$. In this paper we do not consider random moves.

A triplet $\mathcal{G} = (G, D, v_0)$ is called a *positional game form*.

1.2.2 Cycles, Outcomes, and Utility Functions

Let C denote the set of simple (that is, not self-intersecting) directed cycles (di-cycles) in G.

The set of outcomes A can be defined in two ways:

(i) $A = V_T \cup C$, that is, each terminal and each di-cycle is a separate outcome.

(ii) $A = V_T \cup \{C\}$, that is, each terminal is an outcome and all di-cycles (infinite plays) constitute one special outcome $c = \{C\}$.

Case (i) was considered in [2] for two-person ($n = 2$) game forms. In this paper, we analyze case (ii) for n-person games.

Remark 1. *Let us mention that as early as in 1912, Zermelo already considered case (ii) for zero-sum two-person games in his pioneering work [13], where the game of chess was chosen as a basic example. Obviously, the corresponding graph contains di-cycles: one appears whenever a position is repeated in a play. By definition, any infinite play is a draw. In fact, under tournament rules of chess, any player can claim a draw whenever a position appears for the third time in a play. Yet, this difference does not matter to us, since we are going to restrict ourselves to positional (stationary) strategies; see Remark 2 below.*

Note that players can rank outcome c arbitrarily in their preferences. In contrast, in [1] it was assumed that infinite play $c \in A$ is the worst outcome for all players.

1.2.3 Positional Games in Normal Form

A triplet $\mathcal{G} = (G, D, v_0)$ and quadruple $(G, D, v_0, u) = (\mathcal{G}, u)$ are called a *positional form* and a *positional game*, respectively. Positional games can also be represented in *normal form*, as described below. A mapping $x : V \setminus V_T \to E$ that

assigns to every non-terminal position v a move $e \in out(v)$ from this position is called a *situation* or *strategy profile*.

A *strategy* of player $i \in I$ is the restriction $x_i : V_i \to E$ of x to $V_i = D^{-1}(i)$.

Remark 2. *A strategy x_i of a player $i \in I$ is interpreted as a decision plan for every position $v \in V_i$. Note that, by definition, the decision in v can depend only on v itself but not on the preceding positions and moves. In other words, we restrict the players to their pure positional strategies.*

Each strategy profile $x \in X$ uniquely defines a play $p = p(x)$ that starts in v_0 and then follows the moves prescribed by x. This play either ends in a terminal of V_T (p is finite) or results in a cycle, $a(x) = c$ (p is infinite). Thus, we obtain a game form $g(\mathcal{G}) : X \to A$, which is called the *normal form* of $\mathcal{G}$. This game form is standardly represented by an n-dimensional table whose entries are outcomes of $A = V_T \cup \{c\}$; see Figures 1, 6 and 8.

The pair $(g(\mathcal{G}), u)$ is called the *normal form* of a positional game $(\mathcal{G}, u)$.

1.3 On Nash-Solvability of Positional Game Forms

Nash-solvability of positional game forms with **acyclic di-graphs** is well-known. It easily follows from the fact that a Nash equilibrium in such a game can be computed by backward induction [12, 8].

In [2], Nash-solvability of positional game forms was considered for case (i): each di-cycle is a separate outcome. An explicit characterization of Nash-solvability was obtained for the two-person ($n = 2$) game forms whose di-graphs are *bidirected*: $(v', v'') \in E$ if and only if $(v'', v') \in E$.

In [1], Nash-solvability of positional game forms was studied (for a more general class of payoff functions, so-called *additive* or *integral* payoffs), but with the following additional restriction:

(ii') The outcome c, an infinite play, is ranked as the worst one by all players.

Under assumption (ii'), Nash-solvability was proven in three cases:

(a) Two-person games ($n = |I| = 2$);

(b) Games with at most three outcomes ($|A| \leq 3$);

(c) Play-once games, in which each player controls only one position ($|V_i| = 1 \; \forall \; i \in I$).

The proof of Nash-solvability in case (c) is indirect and based on the following simple observation. Let us suppose that this claim fails and consider a minimal counter-example: a play-once game $(\mathcal{G}, u)$ without any Nash equilibria but such that every its proper subgame has one. Then

(*iii*) every finite play p and every di-cycle c in $\mathcal{G}$ intersect.

Indeed, otherwise one could get a smaller counter-example by eliminating di-cycle c (together with all edges incident to c) from $\mathcal{G}$. We have to show that the obtained proper subgame $(\mathcal{G}', u')$ has no Nash equilibrium either.

Let us suppose that there is a Nash equilibrium, x' , and let $p(x')$ be the corresponding play in $\mathcal{G}'$. Then, let us extend x' to a strategy profile x in $\mathcal{G}$ assuming that in every position of c the corresponding player makes a (unique) move along c.

By assumption, x cannot be a Nash equilibrium in the game $(\mathcal{G}, u)$, that is, a player $i \in I$ can improve his result by changing his strategy. Since game $(\mathcal{G}, u)$ is play-once, there is only one way to do this: i must choose a new move (v', v), where $v' \in p(x')$ and $v \in c$. Clearly, the obtained play results in the di-cycle c. Hence, this cannot be an improvement, since c is the worst outcome for all players. Thus, (iii) holds.

Let us remark that both our assumptions were essential in the above proof: if game $(\mathcal{G}, u)$ is not play-once then it could be possible for a player i to enter c, then, later, leave c, and make a profit; if c is not the worst outcome for a player then this player can make a profit by reaching c.

Thus, (iii) must hold for every minimal counter-example. Yet, di-graphs satisfying (iii) have a very special structure; see Lemmas 1-6 of [1]. Recently, Edmonds suggested to call them *Mercator di-graphs*. Although characterizing Mercator graphs is still an open problem, partial results of [1] imply that every positional game form with a Mercator di-graph is Nash-solvable. This results in Nash-solvability for case (c).

It was also conjectured in [1] that Nash-solvability holds in general.

Conjecture 1 ([1])**.** *A positional game is Nash-solvable whenever (ii') holds.*

This Conjecture would be implied by the following statement: *every im-cycle $\mathcal{X} = \{x^1, \dots, x^k\} \subseteq X$ contains a strategy profile x^j such that the corresponding play $p(x^j)$ is infinite.* Indeed, Conjecture 1 would follow, since outcome $c \in A$ being the worst for all players, belongs to no im-cycle. However, the example of Section 2.2 will show that such an approach fails. Nevertheless, Conjecture 1 is not disproved. Moreover, a stronger conjecture was recently suggested by Gimbert and Sørensen, [4]. They assumed that, in case of terminal payoffs, condition (ii') is not needed.

Conjecture 2. *A positional game is Nash-solvable if all cycles form a single outcome.*

They gave a simple and elegant proof for the two-person case. With their permission, we reproduce it in Section 5.

1.4 Restricted Improvement Cycles and Acyclicity

1.4.1 Improvement Cycles in Trees

Kukushkin [9, 10] was the first to consider im-cycles in positional games. He restricted himself to trees and observed that even in this case im-cycles can exist; see example in Figure 1.

However, it is easy to see that unnecessary changes of strategies take place in this im-cycle. For example, let us consider transition from $x^1 = (x_1^1, x_2^2)$ to

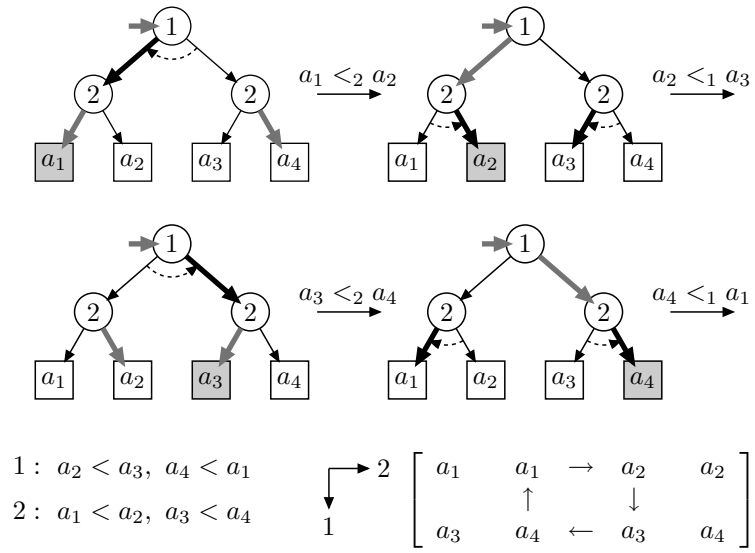


Figure 1: Im-cycle in a tree. Bold arrows indicate chosen moves, and the black ones have changed from the previous strategy profile. The induced preference relations are shown on the left. The matrix on the right shows the normal form of the game.

$x^2 = (x_1^1, x_2^3)$. Player 1 keeps his strategy x_1^1, while 2 substitutes x_2^3 for x_2^2 and gets a profit, since $g(x_1^1, x_2^2) = a_1,\ \ g(x_1^1, x_2^3) = a_2$, and $u_2(a_1) < u_2(a_2)$.

Yet, player 2 switches simultaneously from a_4 to a_3. Obviously, this cannot serve any practical purpose, since the strategy is changed outside the actual play.

In [9], Kukushkin also introduced the concept of *restricted improvements* (ri). In particular, he proved that positional games on trees become ri-acyclic if players are not allowed to change their decisions outside the actual play. For completeness, we sketch his simple and elegant proof in Section 3.1, where we also mention some related results and problems.

Since we consider arbitrary finite di-graphs (not only trees), let us define accurately several types of restrictions for this more general case. This restriction (introduced by Kukushkin) will be called the *inside play restriction*.

1.4.2 Inside Play Restriction

Given a positional game form $\mathcal{G} = (G, D, v_0)$ and strategy profile $x^0 = (x_i^0,\ i \in I) \in X$, let us consider the corresponding play $p_0 = p(x^0)$ and outcome $a_0 = a(x^0) \in A$. This outcome is either a terminal, $a_0 \in V_T$, or a di-cycle, $a_0 = c$.

Let us consider the strategy x_i^0 of a player $i \in I$. He is allowed to change his decision in any position v_1 from p_0. This change will result in a new strategy profile x^1, play $p_1 = p(x^1)$, and outcome $a_1 = a(x^1) \in A$.

Then, player i may proceed, changing his strategy further. Now, he is only allowed to change the decision in any position v_2 that is located *after* v_1 in p_1,

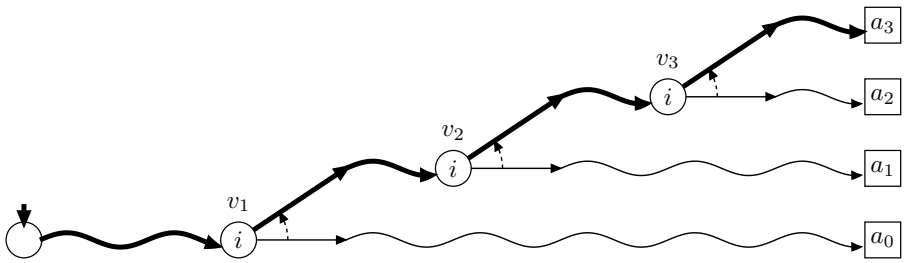


Figure 2: Inside play restriction.

etc., until a position v_m, strategy profile x^m, play $p_m = p(x^m)$, and outcome $a_m = a(x^m) \in A$ appear; see Figure 2, where $m = 3$.

Equivalently, we can say that all positions $v_1, \dots, v_m$ belong to one play.

Let us note that, by construction, obtained plays $\{p_0, p_1, \dots, p_m\}$ are pairwise distinct. In contrast, the corresponding outcomes $\{a_0, a_1, \dots, a_m\}$ can coincide and some of them might be the infinite play outcome $c \in A$.

Whenever the acting player i substitutes the strategy x_i^m, defined above, for the original strategy x_i^0, we say that this is an *inside play deviation*, or in other words, that this change of decision in x satisfies the *inside play restriction*.

It is easy, but important, to notice that this restriction, in fact, does not limit the power of a player. More precisely, if a player i can reach an outcome a_m from x by a deviation then i can also reach a_m by an inside play deviation.

From now on, we will consider only such *inside play restricted* deviations and, in particular, only restricted improvements (ri) and talk about ri-cycles and ri-acyclicity rather than im-cycles and im-acyclicity, respectively.

1.4.3 Types of Improvements

We define the following four types of improvements:

Standard improvement (or just improvement): $u_i(a_m) > u_i(a_0)$;

Strong improvement: $u_i(a_m) > u_i(a_j)$ for $j = 0, 1, \dots, m-1$;

Last step improvement: $u_i(a_m) > u_i(a_{m-1})$;

Best reply (BR) improvement: a_m is the best outcome that player i can reach from x (as we already noticed above, the inside play restriction does not restrict the set of reachable outcomes).

Obviously, each best reply or strong improvement is a standard improvement, and a strong improvement is also a last step improvement. Furthermore, it is easy to verify that no other containments hold between the above four classes. For example, a last step improvement might not be an improvement and vice versa. We will consider ri-cycles and ri-acyclicity specifying in each case a type of improvement from the above list.

Let us note that any type of ri-acyclicity still implies Nash-solvability. Indeed, if a positional game has no NE then for every strategy profile $x \in X$ there

is a player $i \in I$ who can improve x to some other strategy profile $x' \in X$. In particular, i can always choose a strong BR restricted improvement. Since we consider only finite games, such an iterative procedure will result in a strong BR ri-cycle. Equivalently, if we assume that there is no such cycle then the considered game is Nash-solvable; in other words, already strong BR ri-acyclicity implies Nash-solvability.

1.5 Sufficient Conditions for Ri-acyclicity

We start with Kukushkin's result for trees.

Theorem 1 ([9])**.** *Positional games on trees have no restricted standard improvement cycles.*

After trees, it seems natural to consider acyclic di-graphs. We asked Kukushkin [11] whether he had a generalization of Theorem 1 for this case. He had not considered it yet, but shortly thereafter he obtained a result that can be modified as follows.

Theorem 2. *Positional games on acyclic di-graphs have no restricted last step improvement cycles.*

Let us remark that Theorem 1 does not immediately follow from Theorem 2, since a standard improvement might be not a last step improvement. Finally, in case of two players the following statement holds.

Theorem 3. *Two-person positional games have no restricted strong improvement cycles.*

Obviously, Theorem 3 implies Nash-solvability of two-person positional games; see also Section 5 for an independent proof due to Gimbert and Sørensen.

The proofs of Theorems 1, 2 and 3 are given in Sections 3.1, 3.2 and 3.3, respectively.

2 Examples of Ri-cycles; Limits of Theorems 1, 2 and 3

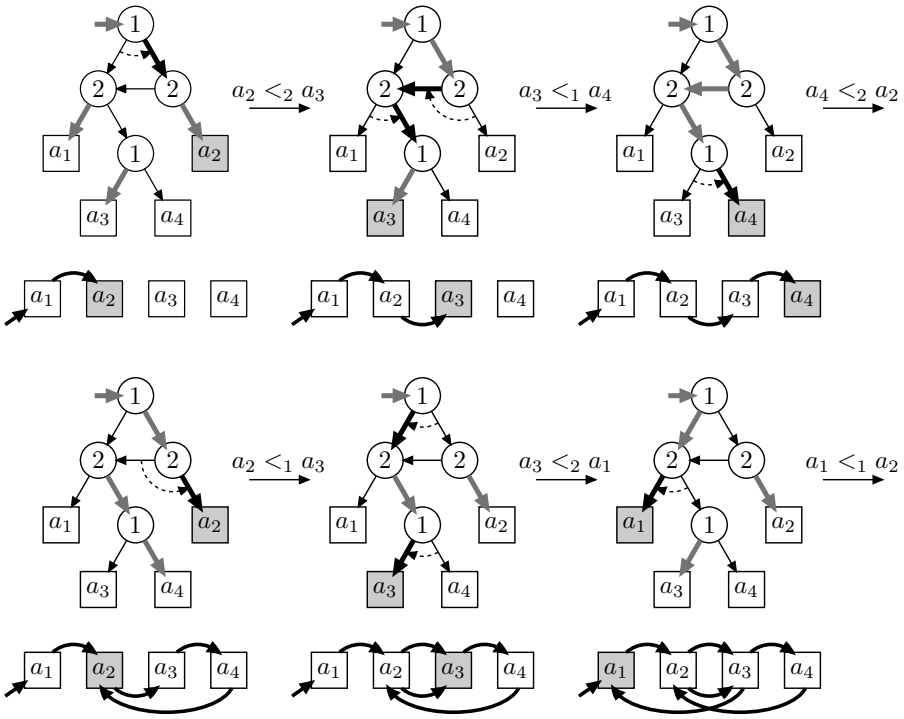
In this paper, we emphasize negative results showing that it is unlikely to strengthen one of the above theorems or obtain other criteria of ri-acyclicity.

2.1 Examples Limiting Theorems 2 and 3

For both Theorems 2 and 3, the specified type of improvement is essential. Indeed, the example in Figure 3 shows that a two-person game on an acyclic di-graph can have a ri-cycle. However, it is not difficult to see that in this ri-cycle, not all improvements are strong and some are not even last step improvements.

Thus, all conditions of Theorems 2 and 3 are essential.

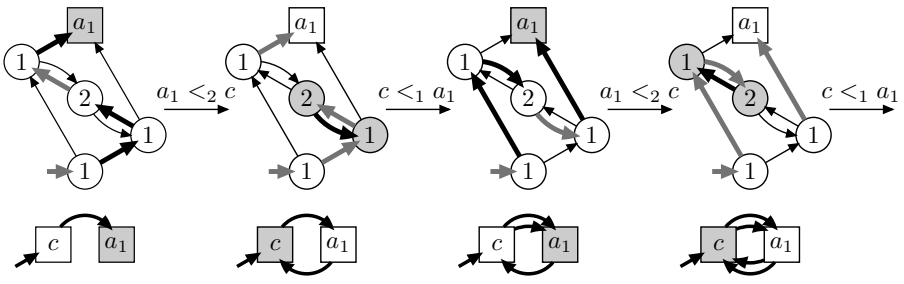
Furthermore, we note that if in Theorem 3 we substitute BR improvement for strong improvement, the modified statement will not hold, see Figure 4.



1 : $a_1 < a_2 < a_3 < a_4$

2 : $a_4 < a_2 < a_3 < a_1$

Figure 3: 2-person ri-cycle in an acyclic di-graph. Beneath each situation is a graph of outcomes with edges defined by the previous improvement steps; these will be of illustrative importance in the proof of Theorem 3.



1 : $c < a_1$

2 : $a_1 < c$

Figure 4: 2-person BR ri-cycle in a di-graph with di-cycles.

By definition, every change of strategy must result in an improvement for the corresponding player. Hence, each such change implies an ordering of the two outcomes; in our figures, it appears as a label on the transition arrow between situations. An entire im-cycle implies a set of inequalities, which must be satisfiable in order to allow a consistent preference profile. Note that it is also sufficient to allow ties and have a partial ordering of the outcomes.

Obviously, these sets of preferences must be consistent, that is, acyclic. Thus, we obtain one more type of cycles and acyclicity: *preference cycles* and preference acyclicity. For example, the ri-cycle in Figure 3 implies:

$u_1(a_1) < u_1(a_2) < u_1(a_3) < u_1(a_4)$ and $u_2(a_4) < u_2(a_2) < u_2(a_3) < u_2(a_1)$.

while the one in Figure 4 implies: $u_1(c) < u_1(a_1),\ u_2(a_1) < u_2(c)$.

2.2 On c-free Ri-cycles

In Section 1.3, we demonstrated that Conjecture 1 on Nash-solvability would follow from the following statement:

> (i) *There are no c-free im-cycles.*

Of course, (i) fails. As we know, im-cycles exist already in trees; see Figure 1. However, let us substitute (i) by the similar but much weaker statement:

> (ii) *Every restricted strong BR ri-cycle contains a strategy profile whose outcome is an infinite play.*

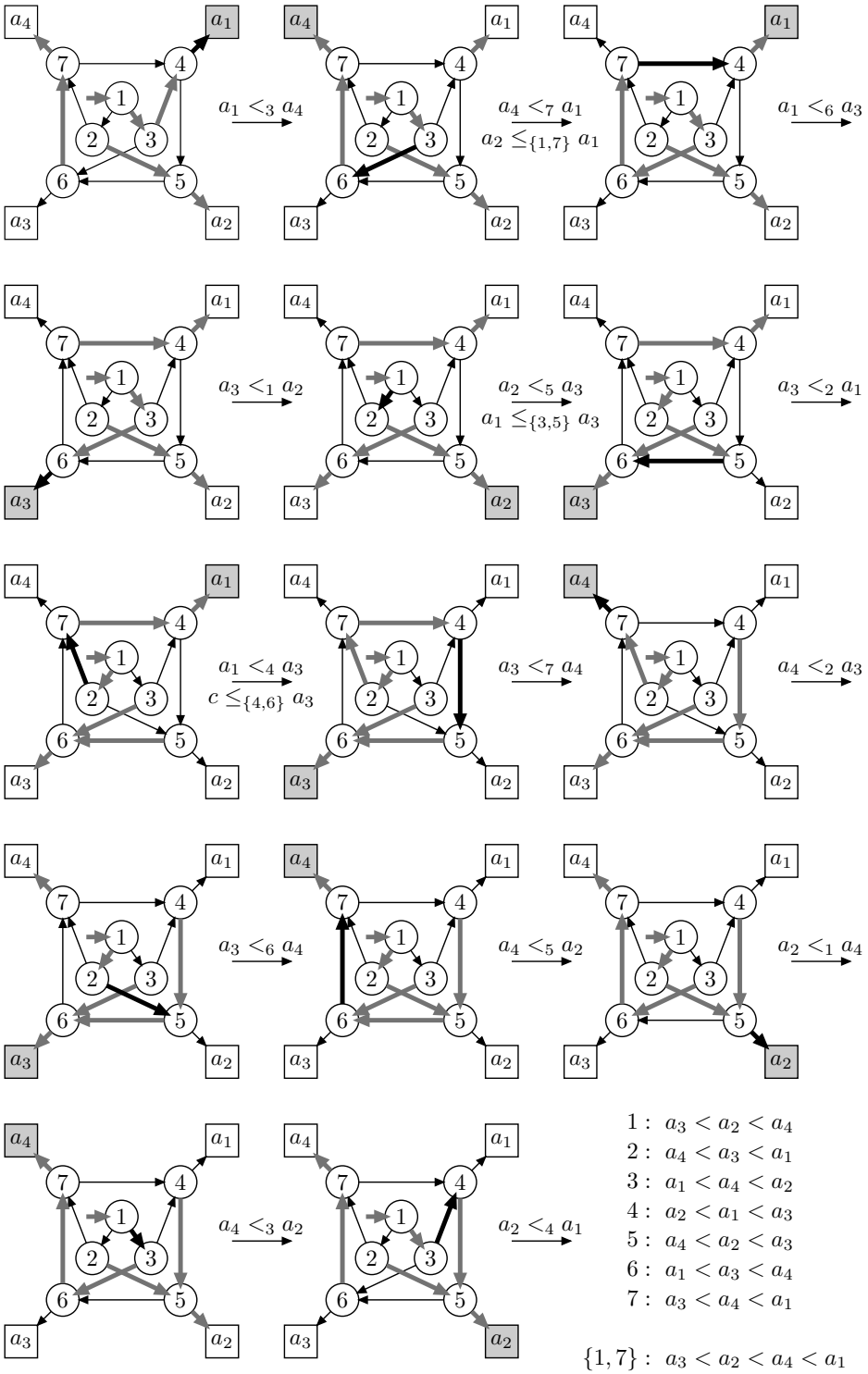
One can derive Conjecture 1 from (ii), as well as from (i). Unfortunately, (ii) also fails. Indeed, let us consider the ri-cycle in Figure 5. This game is play-once: each player controls only one position. Moreover, there are only two possible moves in each position. For this reason, every ri-cycle in this game is BR and strong.

There are seven players ($n = 7$) in this example, yet, by teaming up players in coalitions we can reduce the number of players to four while the improvements remain BR and strong. Indeed, this can be done by forming three coalitions $\{1, 7\}, \{3, 5\}$, $\{4, 6\}$ and merging the preferences of the coalitionists. The required extra constraints on the preferences of the coalitions are also shown in Figure 5.

It is easy to see that inconsistent (i.e., cyclic) preferences appear whenever any three players form a coalition. Hence, the number of coalitions cannot be reduced below 4, and it is, in fact, not possible to form 4 coalitions in any other way while keeping improvements BR and strong.

Obviously, for the two-person case, **(ii)** follows from Theorem 3.

Remark 3. *We should confess that our original motivation fails. It is hardly possible to derive new results on Nash-solvability from ri-acyclicity. Although, ri-acyclicity is much weaker than im-acyclicity, it is still too much stronger than Nash-solvability. By Theorems 3 and 2, ri-acyclicity holds for $n = 2$ and for acyclic di-graphs. Yet, for these two cases Nash-solvability is known.*



$1:\ a_3 < a_2 < a_4$
$2:\ a_4 < a_3 < a_1$
$3:\ a_1 < a_4 < a_2$
$4:\ a_2 < a_1 < a_3$
$5:\ a_4 < a_2 < a_3$
$6:\ a_1 < a_3 < a_4$
$7:\ a_3 < a_4 < a_1$

$\{1,7\}:\ a_3 < a_2 < a_4 < a_1$
$\{2\}:\ a_4 < a_3 < a_1$
$\{3,5\}:\ a_1 < a_4 < a_2 < a_3$
$\{4,6\}:\ a_2 < a_1 < a_3 < a_4,\ c \leq a_3$

Figure 5: c-free strong BR ri-cycle.

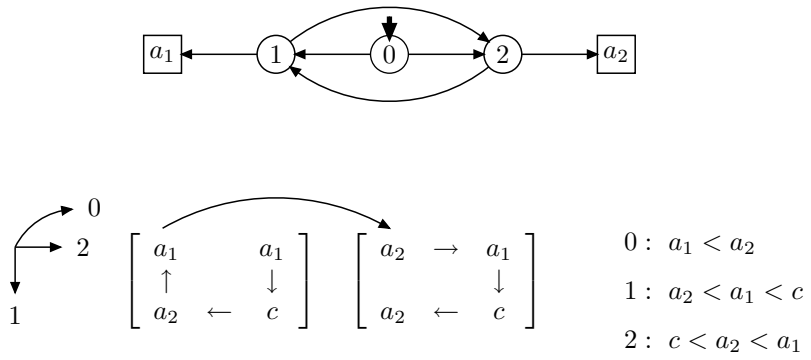


Figure 6: Ri-acyclic flower game form with 3 players.

It is still possible that (ii) (and, hence, Conjecture 1) holds for $n = 3$, too. Strong BR ri-cycles in which the infinite play outcome c occurs do exist for $n = 3$, however. Such an example is provided in Section 2.3.4.

However, ri-acyclicity is of independent (of Nash-solvability) interest. In this paper, we study ri-acyclicity for the case when each terminal is a separate outcome, while all di-cycles form one special outcome. For the alternative case, when each terminal and each di-cycle is a separate outcome, Nash-solvability was considered in [2], while ri-acyclicity was never studied.

2.3 Flower Games: Ri-cycles and Nash-Solvability

2.3.1 Flower Positional Game Forms

A positional game form $\mathcal{G} = (G, D, v_0)$ will be called a *flower* if there is a (chordless) simple di-cycle C in G that contains all positions, except the initial one, v_0, and the terminals, V_T; furthermore, we assume that there are only moves from v_0 to C and from C to V_T; see examples in Figures 6, 7 and 9.

By definition, C is the unique di-cycle in G. Nevertheless, it is enough to make flower games very different from acyclic games; see [1] (where flower games are referred to as St. George games). Here we consider several examples of ri-cycles in flower game forms of 3 and 4 players; see Figures 6, 7 and 9. Let us note that the game forms of Figures 6 and 7 are play-once: each player is in control of one position, that is, $n = |I| = |V \setminus V_T| = 3$ or 4, respectively. In fact, Figure 9 can also be turned into a six-person play-once flower game.

2.3.2 Flower Three-Person Game Form

Positional and normal forms of a three-person flower game are given in Figure 6. This game form is ri-acyclic. Indeed, it is not difficult to verify that an im-cycle in it would result in inconsistent preferences for one of the players. Yet, there is a sequence of 7 restricted improvement steps, that is, a "Hamiltonian improvement path" in the normal form.

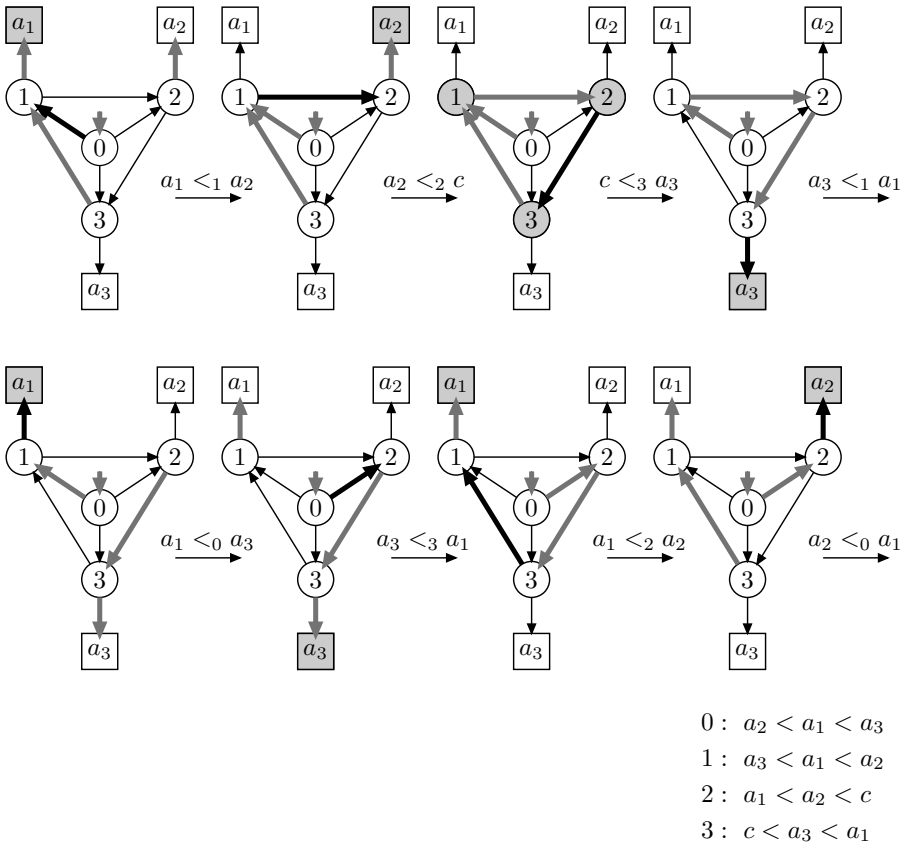


Figure 7: Positional form of a ri-cycle in a flower game form with 4 players.

2.3.3 Flower Four-Person Game Form

Positional and normal forms of a four-person flower game are given in Figures 7 and 8, respectively, where a ri-cycle is shown. Obviously, it is a strong and BR ri-cycle, since there are only two possible moves in every position. However, it contains c.

The number of players can be reduced by forming either the coalition $\{1, 2\}$ or the coalition $\{1, 3\}$. However, in the first case the obtained ri-cycle is not BR, though it is strong, whereas a non-restricted improvement appears in the second case.

Moreover, no c-free ri-cycle can exist in this four-person flower game form. To see this, let us consider the graph of its normal form shown in Figure 8. It is not difficult to verify that, up to isomorphism, there is only one ri-cycle, shown above. All other ri-cycle candidates imply inconsistent preferences; see the second graph in Figure 8.

2.3.4 On BR Ri-cycles in Three-Person Flower Games

In Section 2.2 we gave an example of a c-free strong BR ri-cycle in a four-person game. Yet, the existence of such ri-cycles in three-person games remains open.

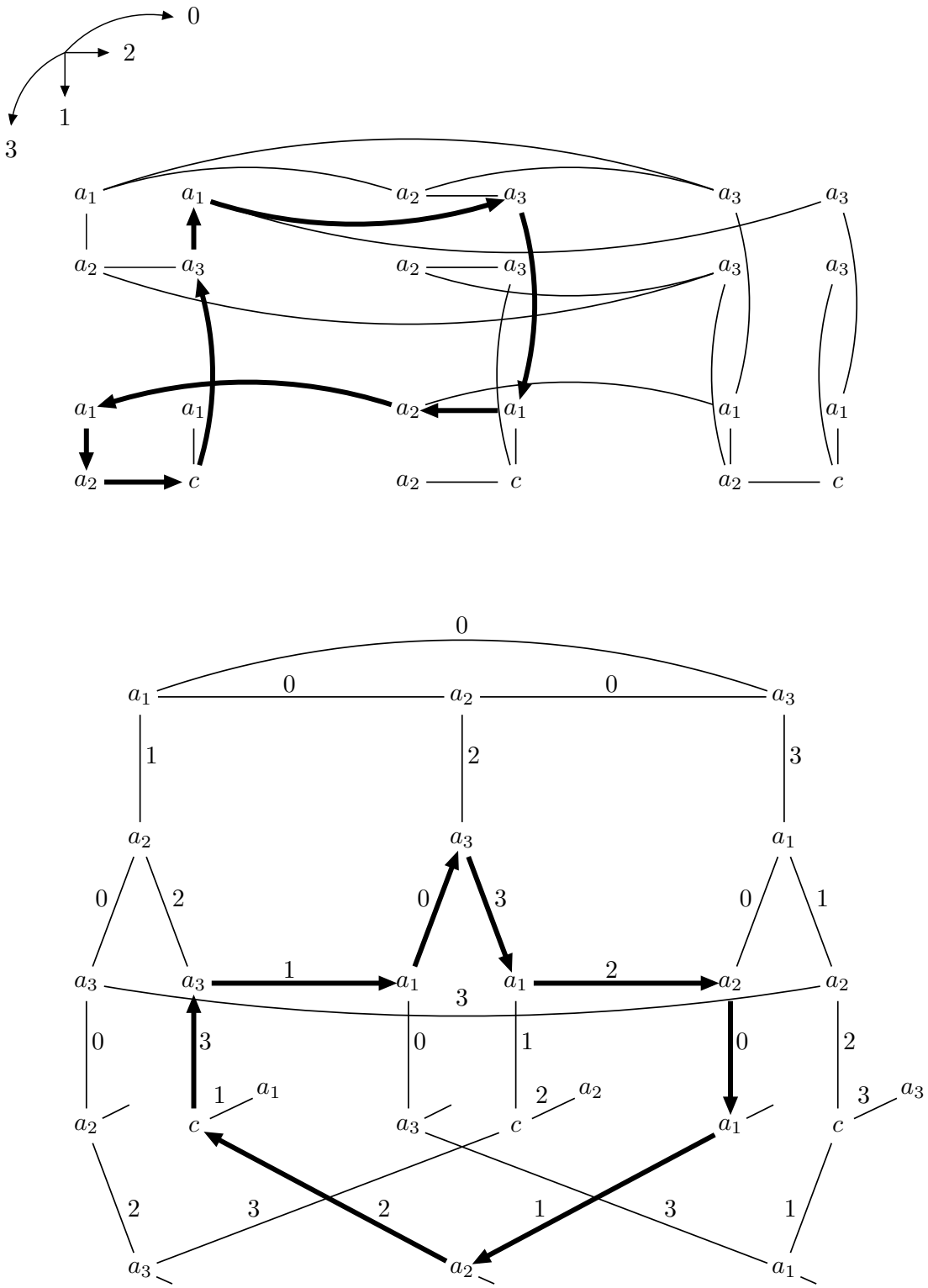


Figure 8: Normal form and unfolded normal form of a ri-cycle in a flower game form with 4 players.

However, a strong BR ri-cycle that contains c can exist already in a three-person flower game; see Figure 9.

2.3.5 Nash-Solvability of Flower Games

In this section we assume without loss of generality that v_0 is controlled by player 1 and that every position v in C has exactly two moves: one along C and the other to a terminal $a = a_v \in V_T$. Indeed, if a player has several terminal moves from one position then, obviously, all but one, which leads to the best terminal, can be eliminated.

We will call positions in C *gates* and, given a strategy profile x, we call gate $v \in C$ *open* (*closed*) if move (v, a_v) is chosen (not chosen) by x.

First, let us consider the simple case when player 1 controls only v_0 and later we will reduce Nash-solvability of general flower games to this special case.

Lemma 1. *Flower games in which player* 1 *controls only* v_0 *are Nash-solvable.*

Proof. Let us assume that there is a move from v_0 to each position of C. In general, the proof will remain almost the same, except for a few extra cases with a similar analysis.

Now, either: (i) for each position $v \in C$ the corresponding player $i = D(v)$ prefers c to $a = a_v$, or (ii) there is a $v' \in C$ such that $i' = D(v')$ prefers $a' = a_{v'}$ to c. If a player controls several such positions then let a' be his best outcome.

In case (i), a strategy profile such that all gates are closed is a NE. In case (ii), the strategy profile where player 1 moves from v_0 to v', the gate v' is open, and all other gates are closed, is a NE. □

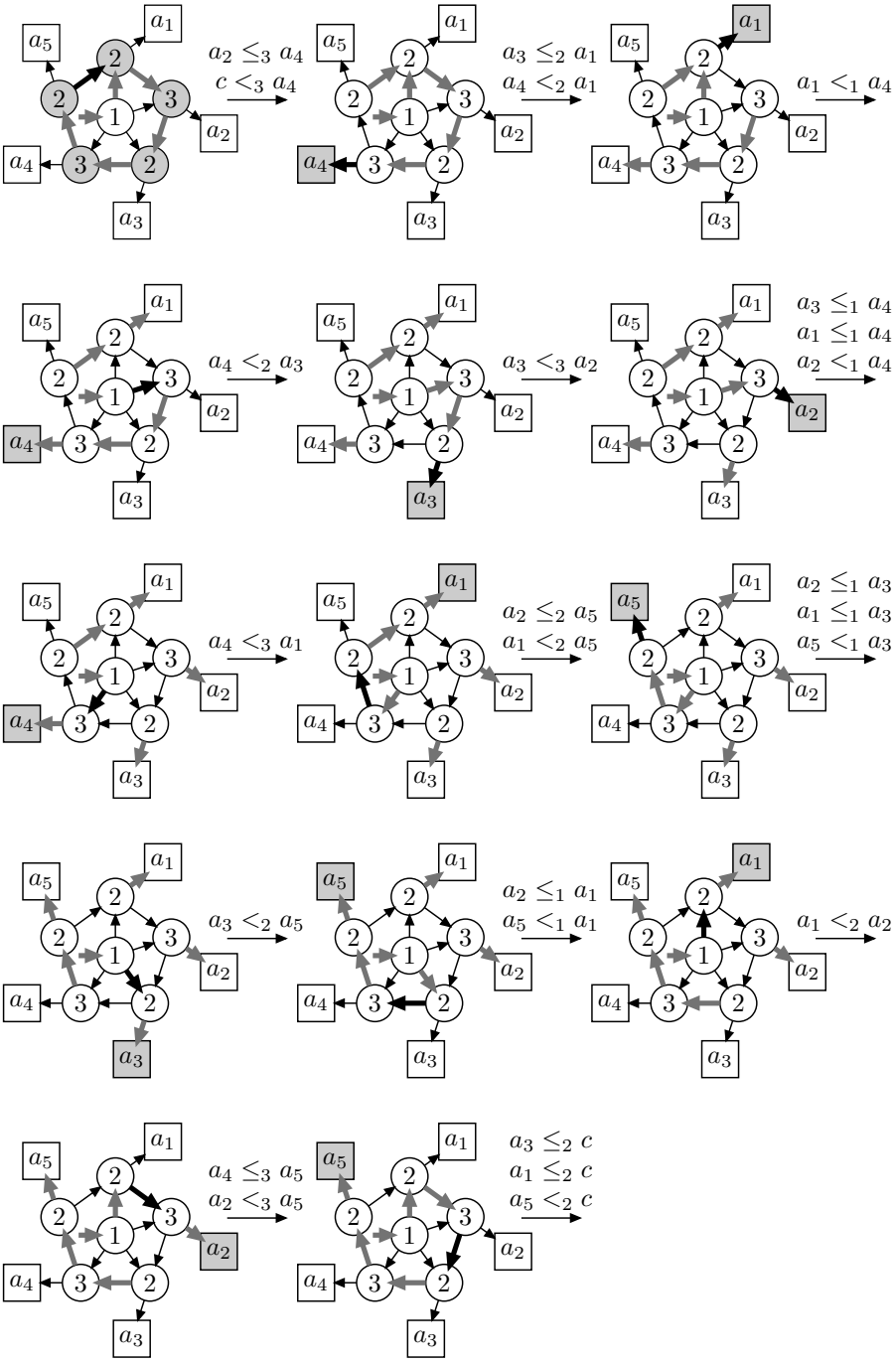
Theorem 4. *Flower games are Nash-solvable.*

Proof. We will give an indirect proof, deriving a contradiction with Lemma 1. Let $(\mathcal{G}, u)$ be a NE-free flower game. Moreover, let us assume that it is *minimal*, that is, a NE appears whenever we delete any move from $\mathcal{G}$. This assumption implies that for each gate v, there is a strategy profile x^1 such that this gate is closed but it is opened by a BR restricted improvement x^2. Since the game is NE-free, there is an infinite sequence $\mathcal{X} = \{x^1, x^2, \ldots\}$ of such BR restricted improvements. Then, it follows from Theorem 2 that gate v will be closed again by a profile $x^k \in \mathcal{X}$. Indeed, if we delete the closing edge (i.e., v remains open), the resulting graph is acyclic.

Now, let us assume that gate v is controlled by player 1. Let v' be the closest predecessor of v in C such that there is a move from v_0 to v'. Opening v, player 1 can at the same time choose the move (v_0, v').

Clearly, until v will be closed again no gate between v' and v in C, including v' itself, will be opened. Indeed, otherwise the corresponding gate could not be closed again by any sequence $\mathcal{X}$ of restricted best replies. Since player 1 already performed a BR, the next one must be performed by another player. However, these players control only the gates between v' and v in C. Hence, one of them will be opened.

Thus, we obtain a contradiction. Indeed, if a NE-free flower game has a gate of player 1, it will never be required to open. By deleting such gates



Feasible total order:

1 : $a_5 < a_2 < a_1 < a_3 < a_4 < c$

2 : $a_4 < a_3 < a_1 < a_2 < a_5 < c$

3 : $c < a_3 < a_2 < a_4 < a_1 < a_5$

Figure 9: Strong BR ri-cycle in a 3-person flower game.

repeatedly one obtains a NE-free flower game that has no gates of player 1. This contradicts Lemma 1. □

3 Proofs of Theorems 1, 2 and 3

3.1 Ri-acyclicity for Trees: Proof of Theorem 1

As we know, im-cycles can exist even for trees (see Section 1.4.1) but ri-cycles cannot. Here we sketch the proof from [9].

Given a (directed) tree $G = (V, E)$ and an n-person positional game $(\mathcal{G}, u) = (G, D, v_0, u)$, let $p_i = \sum_{v \in V_i}(|out(v)| - 1)$ for every player $i \in I = \{1, \ldots, n\}$. It is not difficult to verify that $1 + \sum_{i=1}^{n} p_i = p = |V_T|$.

Let us fix a strategy profile $x = (x_1, \ldots, x_n)$. To every move $e = (v, v')$ which is *not* chosen by x let us assign the outcome $a = a(e, x)$ which x would result in starting from v'. it is easy to see that these outcomes together with $a(x)$ form a partition of V_T.

Given a player $i \in I$, let us assign p_i numbers $u_i(a(e, x))$ for all $e = (v, v')$ not chosen by x_i, where $v \in V_i$. Let us order these numbers in monotone non-increasing order and denote the obtained p_i-dimensional vector $y_i(x)$.

Let player $i \in I$ substitute a restricted improvement x_i' for x_i; see Section 1.4.2. The new strategy profile x' results in an outcome $a_k \in A = V_T$ which is strictly better for i than the former outcome $a_0 = a(x)$. Let us consider vectors $y_j(x)$ and $y_j(x')$ for all $j \in I$. It is not difficult to verify that these two vectors are equal for each $j \in I$, except $j = i$, while $y_i(x)$ and $y_i(x')$, for the acting player i, differ by only one number: $u_i(a_k)$ in $y_i(x')$ substitutes for $u_i(a_0)$ in $y_i(x)$. The new number is strictly larger than the old one, because, by assumption of Theorem 1, x_i' is an improvement with respect to x_i for player i. Thus, vectors y_j for all $j \neq i$ remain unchanged, while y_i becomes strictly larger. Hence, no ri-cycle can appear. □

An interesting question: what is the length of the longest sequence of restricted improvements? Given $n = |I|$, $p = |A|$, and p_i such that $\sum_{i=1}^{n} p_i = p - 1 \geq n \geq 1$, the above proof of Theorem 1 implies the following upper bound: $\sum_{i=1}^{n} p_i(p - p_i)$. It would also be interesting to get an example with a high lower bound.

3.2 Last Step Ri-acyclicity for Acyclic Di-graphs: Proof of Theorem 2

Given a positional game $(\mathcal{G}, u) = (G, D, v_0, u)$ whose di-graph $G = (V, E)$ is acyclic, let us order positions of V so that $v < v'$ whenever there is a directed path from v to v'. To do so, let us assign to each position $v \in V$ the length of a longest path from v_0 to v and then order arbitrarily positions with equal numbers.

Given a strategy profile x, let us, for every $i \in I$, assign to each position $v \in V_i$ the outcome $a(v, x)$ which x would result in starting from v and the number $u_i(a(v, x))$. These numbers form a $|V \setminus V_T|$-dimensional vector $y(x)$ whose coordinates are assigned to positions $v \in V \setminus V_T$. Since these positions

are ordered, we can introduce the inverse lexicographic order over such vectors y.

Let a player $i \in I$ choose a last step ri-deviation x_i' from x_i. Then, $y(x') > y(x)$, since the last changed coordinate increased: $u_i(a_k) > u_i(a_{k-1})$. Hence, no last step ri-cycle can exist. □

3.3 Strong Ri-acyclicity of Two-Person Games: Proof of Theorem 3

Let us consider a two-person positional game $\mathcal{G} = (G, D, v_0, u)$ and a strategy profile x such that in the resulting play $p = p(x)$ the terminal move (v, a) belongs to a player $i \in I$. Then, a strong improvement x_i' results in a terminal $a' = p(x')$ such that $u_i(a') > u_i(a)$. (This holds for n-person games, as well.)

Given a *strong* ri-cycle $\mathcal{X} = \{x^1, \ldots, x^k\} \in X$, let us assume, without any loss of generality, that the game (G, D, v_0, u) is *minimal* with respect to $\mathcal{X}$, that is, the ri-cycle $\mathcal{X}$ is broken by eliminating any move from $\mathcal{G}$. Furthermore, let $\mathcal{A}(\mathcal{X})$ denote the set of the corresponding outcomes: $A(\mathcal{X}) = \{a(x^j),\ j = 1, \ldots, k\}$. Note that several of the outcomes of the strategy profiles may be the same and that $A(\mathcal{X})$ may contain $c \in A$ (infinite play).

Let us introduce the directed multigraph $\mathcal{E} = \mathcal{E}(\mathcal{X})$ whose vertex-set is $A(\mathcal{X})$ and the directed edges are k pairs (a_j, a_{j+1}), where $a_j = a(x^j),\ \ j = 1, \ldots, k$, and $k + 1 = 1$. It is easy to see that $\mathcal{E}$ is *Eulerian*, i.e., $\mathcal{E}$ is strongly connected and for each vertex its in-degree and out-degree are equal. An example of this construction is shown in Figure 3.

Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be the subgraphs of $\mathcal{E}$ induced by the edges corresponding to deviations of players 1 and 2, respectively. Then, $\mathcal{E}_1$ and $\mathcal{E}_2$ are acyclic, since a cycle would imply an inconsistent preference relation. In the example of Figure 3, the edges are partitioned accordingly (above and below the vertices), and the subgraphs are indeed acyclic.

Hence, there is a vertex a^1 whose out-degree in $\mathcal{E}_1$ and in-degree in $\mathcal{E}_2$ both equal 0. In fact, an outcome $a^1 \in A(\mathcal{X})$ most preferred by player 1 must have this property. (We do not exclude ties in preferences; if there are several best outcomes of player 1 then a^1 can be any of them.) Similarly, we define a vertex a^2 whose in-degree in $\mathcal{E}_1$ and out-degree in $\mathcal{E}_2$ both equal 0.

Let us remark that either a^1 or a^2 might be equal to c, yet, not both. Thus, without loss of generality, we can assume that a^1 is a terminal outcome.

Obviously, a player, 1 or 2, has a move to a^1. If 1 has such a move then it cannot be improved in $\mathcal{X}$, since $u_1(a_j) \le u_1(a^1)$ for all $j = 1, \ldots, k$ and $\mathcal{X}$ is a strong ri-cycle. Let us also recall that a^1 has no incoming edges in $\mathcal{E}_2$. Hence, in $\mathcal{X}$, player 2 never makes an improvement that results in a^1. In other words, a player who has a move to a^1 will make it either always, player 1, or never, player 2. In both cases we obtain a contradiction with the minimality of di-graph $\mathcal{G}$. □

4 Laziness Restriction

In addition to the inside play restriction, let us consider the following closely related but stronger restriction.

Let player i substitute strategy x_i' for x_i to get a new outcome $a' = a(x')$ instead of $a = a(x)$. We call such a deviation *lazy*, or say that it satisfies the *laziness restriction*, if it minimizes the number of positions in which player i changes the decision to reach a'.

Let us note that the corresponding strategy x_i' might not be unique.

Obviously, each lazy deviation satisfies the inside play restriction.

Furthermore, if a lazy deviation is an improvement, $u_i(a) < u_i(a')$, then this improvement is strong.

Proposition 1. *Given a strategy profile x, a target outcome $a' \in A$, and a player $i \in I$, the problem of finding a lazy deviation from x_i to x_i' such that $a(x') = a'$ (and x' is obtained from x by substituting x_i' for x_i) reduces to the shortest directed path problem.*

Proof. Let us assign a length $d(e)$ to each directed edge $e \in E$ as follows: $d(e) = 0$ if move e is prescribed by x, $d(e) = 1$ for every other possible move of the acting player i, and $d(e) = \infty$ for all other edges. Then let us consider two cases: (i) $a' \in V_T$ is a terminal and (ii) $a' = c$.

In case (i), a shortest directed path from v_0 to a' defines a desired x_i', and vice versa. Case (ii), $a' = c$, is a little more complicated.

First, for every directed edge $e = (v, v') \in E$, let us find a shortest directed cycle C_e that contains e and its length d_e. This problem is easily reducible to the shortest directed path problem, too. The following reduction works for an arbitrary weighted di-graph $G = (V, E)$. Given a directed edge $e = (v, v') \in E$, let us find a shortest directed path from v' to v. In case of non-negative weights, this can be done by Dijkstra's algorithm.

Then, it is also easy to find a shortest di-cycle C_v through a given vertex $v \in V$ and its length d_v; obviously, $d_v = \min_{v' \in V}(d_e \mid e = (v, v'))$.

Then, let us apply Dijkstra's algorithm again to find a shortest path p_v from v_0 to every vertex $v \in V$ and its length d_v^0.

Finally, let us find a vertex v^* in which $\min_{v \in V}(d_v^0 + d_v)$ is reached. It is clear that the corresponding shortest directed path p_{v^*} and di-cycle C_{v^*} define the desired new strategy x_i'. □

5 Nash-Solvability of Two-Person Game Forms

If $n = 2$ and $c \in A$ is the worst outcome for both players, Nash-solvability was proven in [1]. In fact, the last assumption is not necessary: even if outcome c is ranked by two players arbitrarily, Nash-solvability still holds. This observation was recently made by Gimbert and Sørensen [4].

A two-person game form g is called:

Nash-solvable if for every utility function $u : \{1, 2\} \times A \to \mathbb{R}$ the obtained game (g, u) has a Nash equilibrium.

zero-sum-solvable if for each zero-sum utility function ($u_1(a) + u_2(a) = 0$ for all $a \in A$) the obtained zero-sum game (g, u) has a Nash equilibrium, which is called a saddle point for zero-sum games.

$\pm$*-solvable* if zero-sum solvability holds for each u that takes only values: $+1$ and -1.

Necessary and sufficient conditions for zero-sum solvability were obtained by Edmonds and Fulkerson [3] in 1970; see also [5]. Somewhat surprisingly, these conditions remain necessary and sufficient for $\pm$-solvability and for Nash-solvability, as well; in other words, all three above types of solvability are equivalent, in case of two-person game forms [6]; see also [7] and Appendix 1 of [2].

Proposition 2. *Each two-person positional game form in which all di-cycles form one outcome is Nash-solvable.*

Proof. Let $\mathcal{G} = (G, D, v_0, u)$ be a two-person zero-sum positional game, where $u : I \times A \to \{-1, +1\}$ is a zero-sum ± 1 utility function. Let $A_i \subseteq A$ denote the outcomes winning for player $i \in I = \{1, 2\}$. Without any loss of generality we can assume that $c \in A_1$, that is, $u_1(c) = 1$, while $u_2(c) = -1$. Let $V^2 \subseteq V$ denote the set of positions in which player 2 can enforce a terminal from A_2. Then, obviously, player 2 wins whenever $v_0 \in V^2$. Let us prove that player 1 wins otherwise, when $v_0 \in V^1 = V \setminus V^2$.

Indeed, if $v \in V^1 \cap V_2$ then $v' \in V^1$ for every move (v, v') of player 2; if $v \in V^1 \cap V_1$ then player 1 has a move (v, v') such that $v' \in V_1$. Let player 1 choose such a move for every position $v \in V^1 \cap V_1$ and an arbitrary move in each remaining position $v \in V^2 \cap V_1$. This rule defines a strategy x_1. Let us fix an arbitrary strategy x_2 of player 2 and consider the profile $x = (x_1, x_2)$. Obviously, the play $p(x)$ cannot come to V_2 if $v_0 \in V_1$. Hence, for the outcome $a = a(x)$ we have: either $a \in V^1$ or $a = c$. In both cases player 1 wins. Thus, the game is Nash-solvable. □

Let us recall that this result also follows immediately from Theorem 3.

Finally, let us briefly consider a refinement of the Nash equilibrium concept, the so-called *subgame perfect* equilibrium, where a strategy profile is an equilibrium regardless of the choice of starting position.

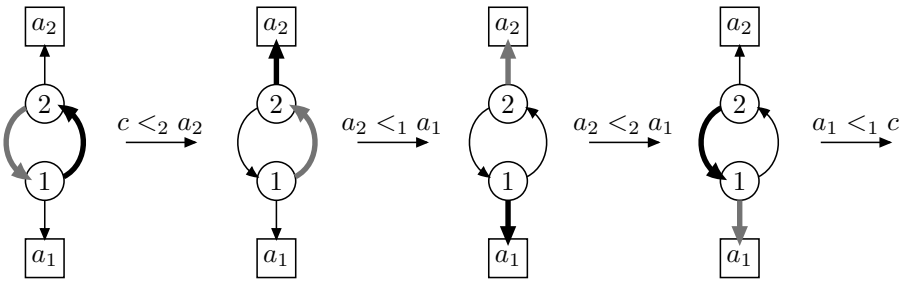
It is not difficult to see that already for two-person games a Nash equilibrium can be unique but not subgame perfect. Let us consider the example in Figure 10. There are only four different strategy profiles and for all of them there is a choice of starting position for which the profile is not an equilibrium.

6 Conclusions and Open Problems

Nash-solvability of n-person positional games (in which all di-cycles form a single outcome) holds for $n = 2$ and remains an open problem for $n > 2$.

For $n = 2$, we prove strong ri-acyclicity, which implies Nash-solvability. Computing Nash equilibria efficiently is another interesting issue for further investigation.

For $n \geq 4$ there are examples of best reply strong c-free ri-cycles. Yet, it remains open whether such c-free examples exist for $n = 3$.



1 : $a_2 < a_1 < c$

2 : $c < a_2 < a_1$

Figure 10: Two-person game with no subgame perfect positional strategies. The improvements do not obey any inside play restriction, since there is no fixed starting position.

Acknowledgements. We are thankful to Gimbert, Kukushkin, and Sørensen for helpful discussions.

References

[1] E. Boros and V. Gurvich, On Nash-solvability in pure strategies of finite games with perfect information which may have cycles. *Math. Social Sciences* 46, 2003.

[2] E. Boros, V. Gurvich, K. Makino, and Wei Shao, Nash-solvable bidirected cyclic two-person game forms, Rutcor Research Report 26-2007 and DIMACS Technical Report 2008-13, Rutgers University.

[3] J. Edmonds and D.R. Fulkerson, Bottleneck Extrema, RM-5375-PR, The Rand Corporation, Santa Monica, Ca., Jan. 1968; *J. Combin. Theory* 8:299–306, 1970.

[4] H. Gimbert and T.B. Sørensen, Private communications, July 2008.

[5] V. Gurvich, To theory of multi-step games *USSR Comput. Math. and Math. Phys.* 13(6):143–161, 1973.

[6] V. Gurvich, Solution of positional games in pure strategies, *USSR Comput. Math. and Math. Phys.* 15(2):74–87, 1975.

[7] V. Gurvich, Equilibrium in pure strategies, *Soviet Mathematics Doklady* 38(3):597-602, 1988.

[8] H. Kuhn, Extensive form games and the problem of information, in H. Kuhn and A. Tucker (eds.), *Contributions to the theory of games, Vol. 2*, pages 193–216, 1953.

[9] N.S. Kukushkin, Perfect information and potential games, *Games and Economic Behavior* 38:306–317, 2002.

[10] N.S. Kukushkin, Acyclicity of improvements in finite game forms, Manuscript, `http://www.ccas.ru/mmes/mmeda/ququ/GF.pdf`, 2009.

[11] N. S. Kukushkin, Private communications, August 2008.

[12] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1944.

[13] E. Zermelo, Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels, *Proc. 5th Int. Cong. Math. Cambridge 1912* , Vol. II, pages 501–504, Cambridge University Press, 1913.

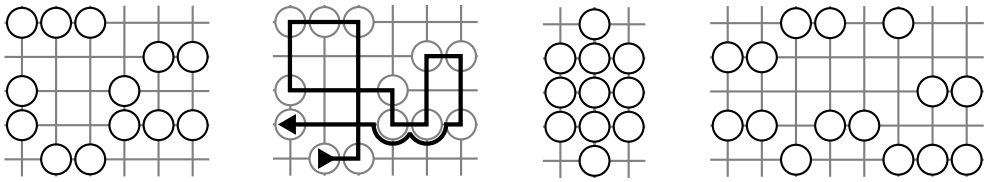
Paper 4

Hiroimono is NP-Complete

Daniel Andersson

Abstract. In a Hiroimono puzzle, one must collect a set of stones from a square grid, moving along grid lines, picking up stones as one encounters them, and changing direction only when one picks up a stone. We show that deciding the solvability of such puzzles is NP-complete.

This is a reformatted version of my paper in *Proceedings of the Fourth International Conference on Fun with Algorithms* (FUN 2007), LNCS 4475.



(a) (b) (c) (d)

Figure 1: (a) A Hiroimono puzzle. (b) A solution to (a). (c) An unsolvable puzzle. (d) An exercise.

1 Introduction

Hiroimono (拾い物, "things picked up") is an ancient Japanese class of tour puzzles. In a Hiroimono puzzle, we are given a square grid with stones placed at some grid points, and our task is to move along the grid lines and collect all the stones, while respecting the following rules:

1. We may start at any stone.
2. When a stone is encountered, we must pick it up.
3. We may change direction only when we pick up a stone.
4. We must not make 180° turns.

Figure 1 shows some small example puzzles.

Although it is more than half a millennium old [2], Hiroimono, also known as Goishi Hiroi (碁石ひろい), appears in magazines, newspapers, and the World Puzzle Championship. Many other popular games and puzzles have been studied from a complexity-theoretic point of view and proved to give rise to hard computational problems, e.g. Tetris [4], Minesweeper [6], Sokoban [3], and Sudoku (also known as Number Place) [7]. We shall see that this is also the case for Hiroimono.

We will show that deciding the solvability of a given Hiroimono puzzle is NP-complete and that specifying a starting stone (a common variation) and/or allowing 180° turns (surprisingly uncommon) does not change this fact.

Definition 1. HIROIMONO *is the problem of deciding for a given nonempty list of distinct integer points representing a set of stones on the Cartesian grid, whether the corresponding Hiroimono puzzle is solvable under rules 1–4. The definition of* START-HIROIMONO *is the same, except that it replaces rule 1 with a rule stating that we must start at the first stone in the given list. Finally,* 180-HIROIMONO *and* 180-START-HIROIMONO *are derived from* HIROIMONO *and* START-HIROIMONO*, respectively, by lifting rule 4.*

Theorem 1. *All problems in Definition 1 are NP-complete.*

These problems obviously belong to NP. To show their hardness, we will construct a reduction from 3-SAT [5] to all four of them.

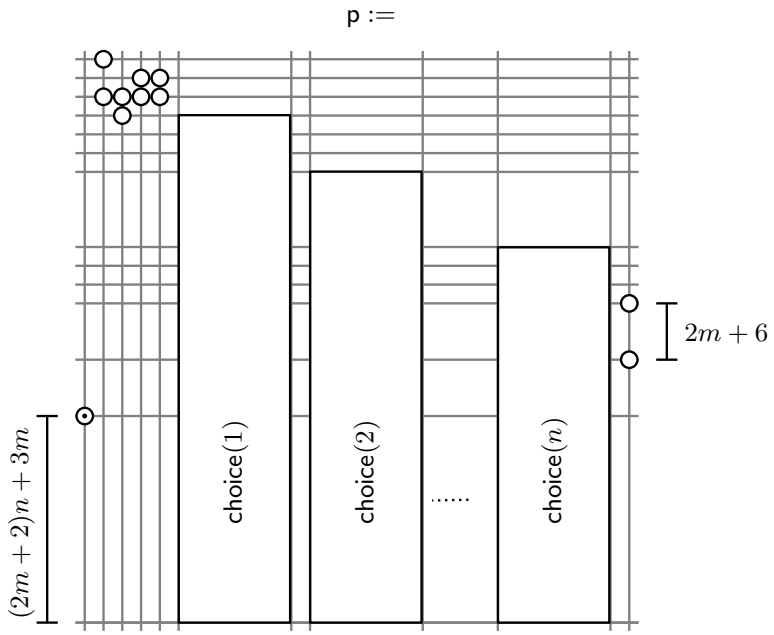


Figure 2: The puzzle p corresponding to the formula ϕ. Although formally, problem instances are ordered lists of integer points, we leave out irrelevant details like orientation, absolute position, and ordering after the first stone ⊙.

2 Reduction

Suppose that we are given as input a CNF formula $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ with variables $x_1, x_2, \ldots, x_n$ and with three literals in each clause. We output the puzzle p defined in Fig. 2–4. Figure 5 shows an example.

3 Correctness

From Definition 1, it follows that

$$\text{START-HIROIMONO} \subseteq \begin{matrix}\text{HIROIMONO} \\ \text{180-START-HIROIMONO}\end{matrix} \subseteq \text{180-HIROIMONO.}$$

Thus, to prove that the map $\phi \mapsto$ p from the previous section is indeed a correct reduction from 3-SAT to each of the four problems above, it suffices to show that $\phi \in$ 3-SAT $\Rightarrow$ p $\in$ START-HIROIMONO and p $\in$ 180-HIROIMONO $\Rightarrow \phi \in$ 3-SAT.

3.1 Satisfiability Implies Solvability

Suppose that ϕ has a satisfying truth assignment t^*. We will solve p in two stages. First, we start at the leftmost stone ⊙ and go to the upper rightmost stone along the path R(t^*), where we for any truth assignment t, define R(t) as shown in Fig. 6–8.

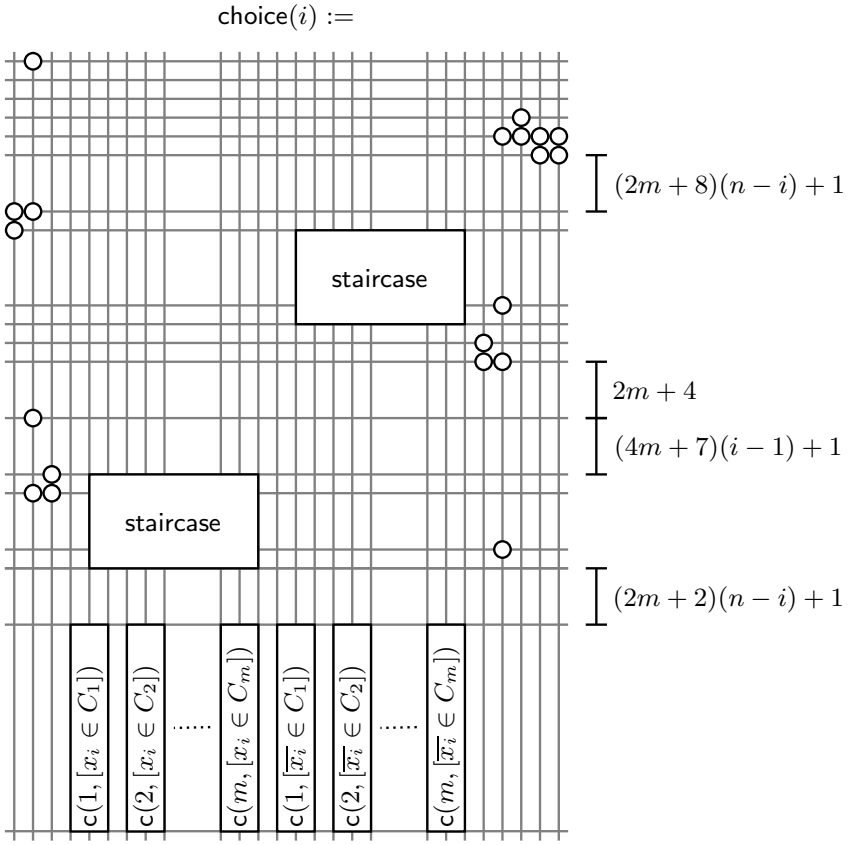


Figure 3: The definition of choice(i), representing the variable x_i. The two staircase-components represent the possible truth values, and the c-components below them indicate the occurrence of the corresponding literals in each clause.

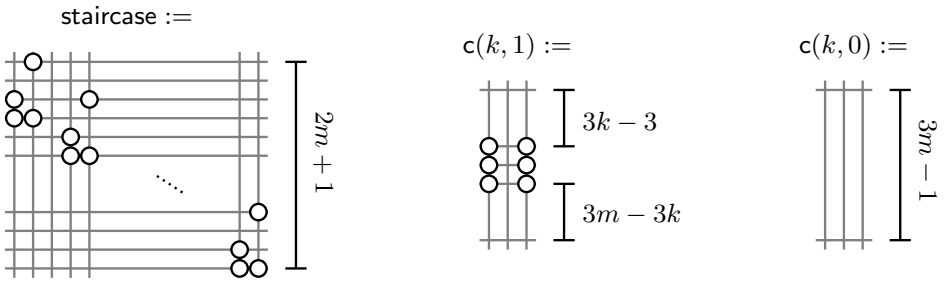


Figure 4: The definition of staircase, consisting of m "steps", and the c-components. Note that for any fixed k, all c$(k, 1)$-components in p, which together represent C_k, are horizontally aligned.

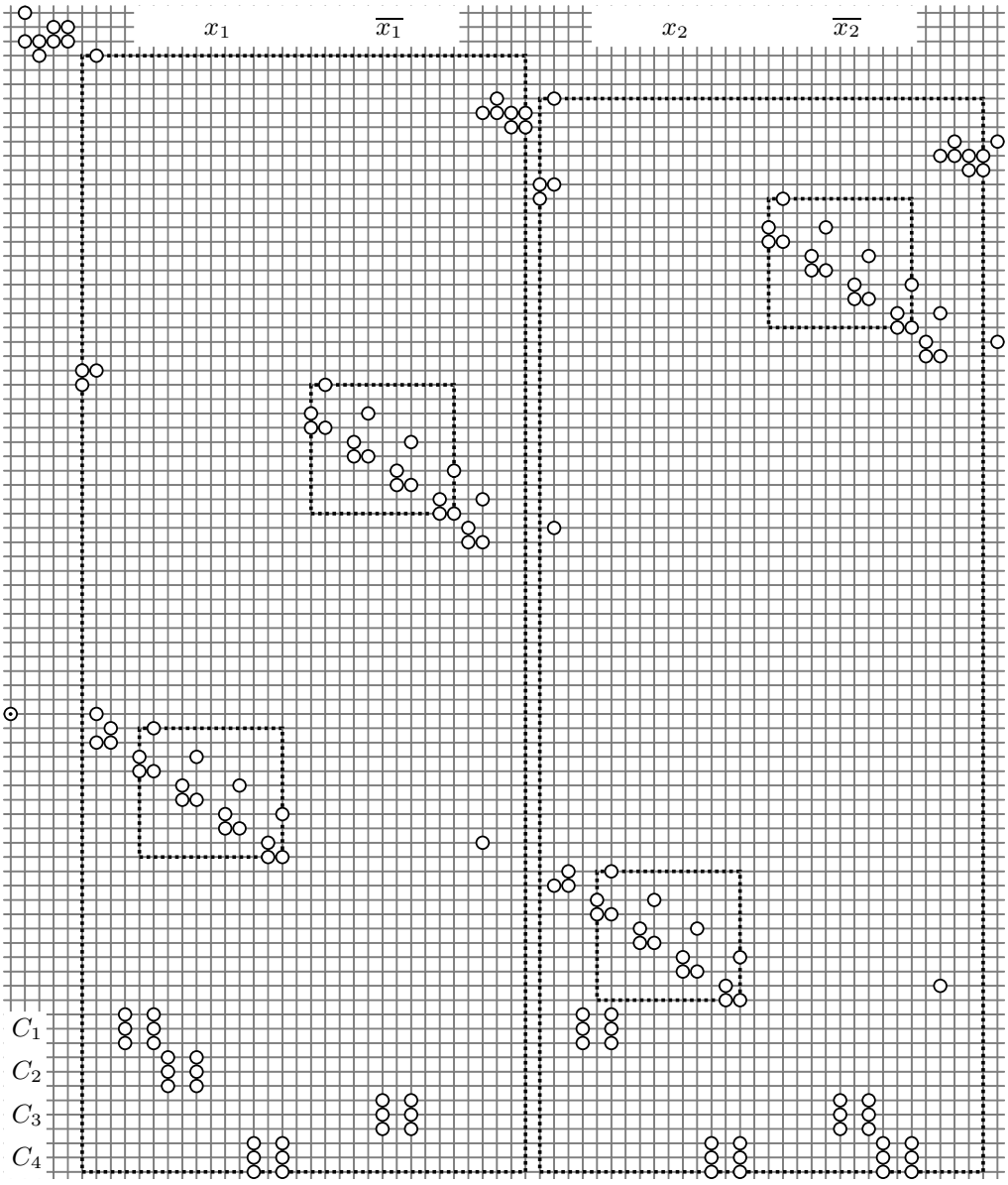


Figure 5: If $\phi = (x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee x_1 \vee x_1) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (x_1 \vee x_2 \vee \overline{x_2})$, this is p. Labels indicate the encoding of clauses, and dotted boxes indicate choice(1), choice(2), and staircase-components. The implementation that generated this example is accessible online [1].

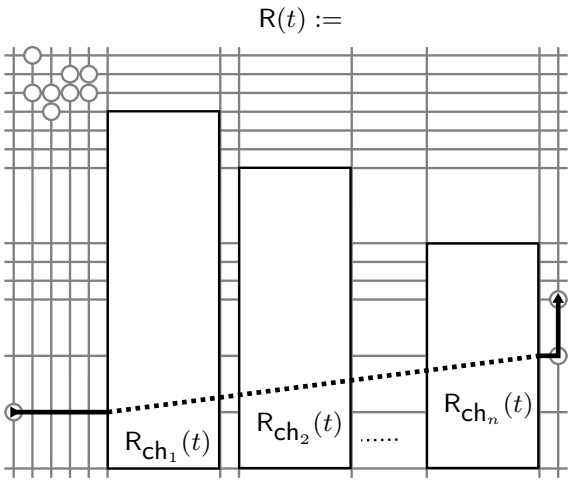


Figure 6: The path $\mathsf{R}(t)$, which, if t satisfies ϕ, is the first stage of solving p.

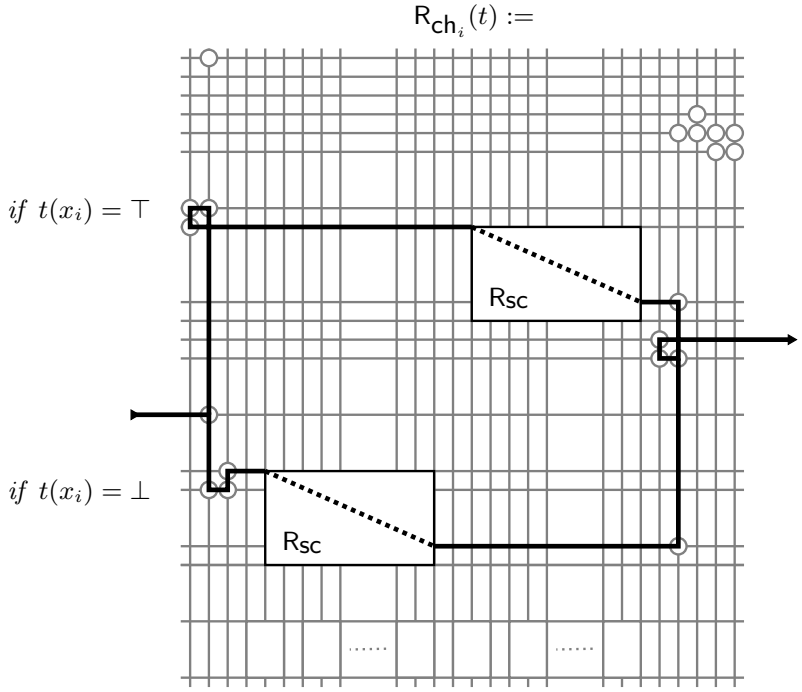


Figure 7: Assigning a truth value by choosing the upper or lower staircase.

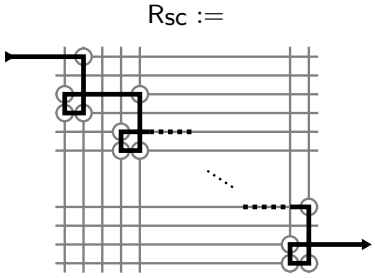


Figure 8: Descending a staircase.

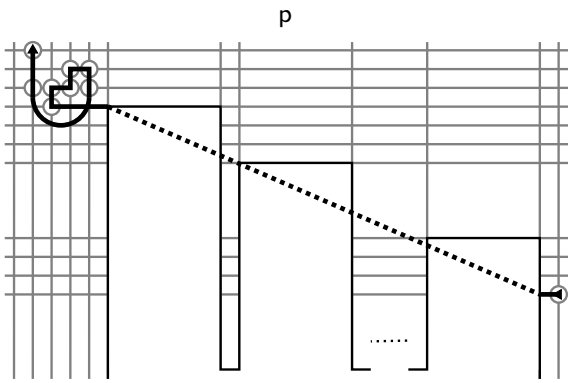


Figure 9: The second stage of solving p.

Definition 2. *Two stones on the same grid line are called* neighbors.

By the construction of p and R, we have the following:

Lemma 1. *For any t and k, after* $\mathsf{R}(t)$*, there is a stone in a* $\mathsf{c}(k,1)$*-component with a neighbor in a* staircase*-component if and only if t satisfies C_k.*

In the second stage, we go back through the choice-components as shown in Fig. 9 and 10. We climb each remaining staircase by performing R_{sc} backwards, but whenever possible, we use the first matching alternative in Fig. 11 to "collect a clause". By Lemma 1, we can collect all clauses. See Fig. 12 for an example.

Since this two-stage solution starts from the first stone ⊙ and does not make 180° turns, it witnesses that $\mathsf{p} \in$ START-HIROIMONO.

3.2 Solvability Implies Satisfiability

Suppose that $\mathsf{p} \in$ 180-HIROIMONO, and let s be any solution witnessing this (assuming neither that s starts at the leftmost stone nor that it avoids 180° turns). Now consider what happens as we solve p using s. Note that since the topmost stone and the leftmost one each have only one neighbor, s must start at one of these and end at the other. We will generalize this type of reasoning to *sets* of stones.

Definition 3. *A* situation *is a set of remaining stones and a current position. A* dead end D *is a nonempty subset of the remaining stones such that:*

- *There is at most one remaining stone not in* D *that has a neighbor in* D.
- *No stone in* D *is on the same grid line as the current position.*

A hopeless *situation is one with two disjoint dead ends.*

Since the stones in a dead end must be the very last ones picked up, a solution can never create a hopeless situation. If we start at the topmost stone, then we will after collecting at most four stones find ourselves in a hopeless situation, as is illustrated in Fig. 13. Therefore, s must start at the leftmost stone and end at the topmost one.

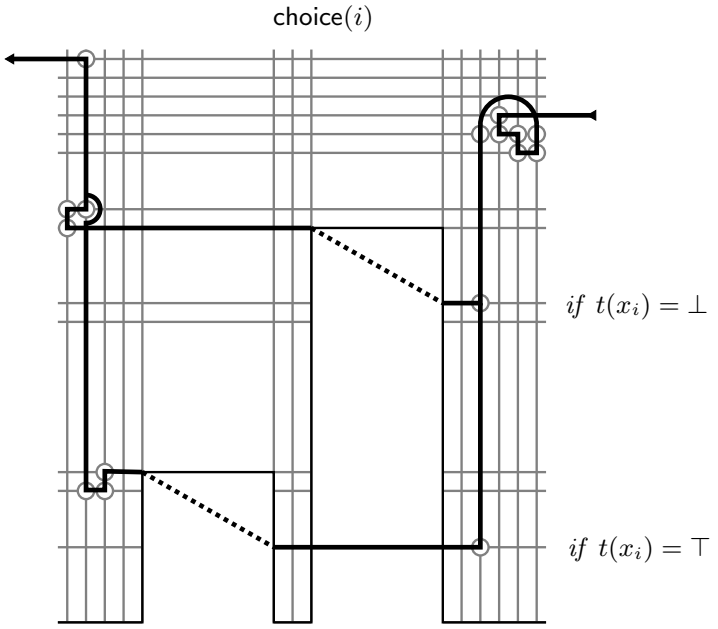


Figure 10: In the second stage, the remaining staircase-component in choice(i) is collected.

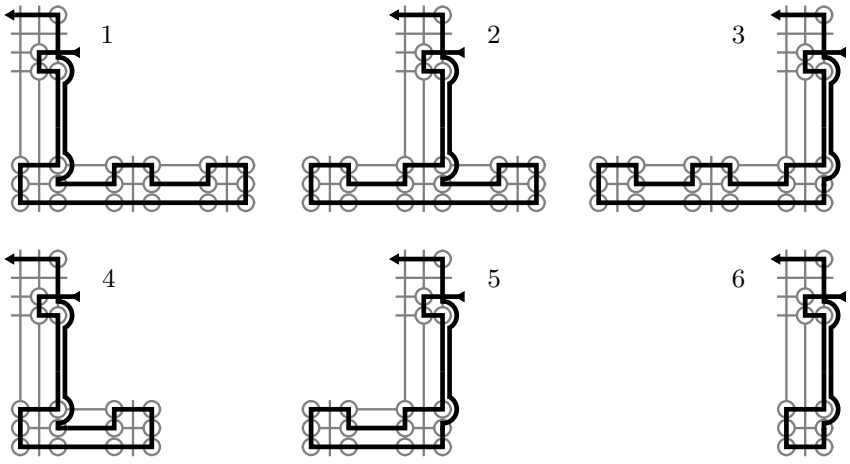


Figure 11: Six different ways to "collect a clause" when climbing a step in a staircase-component.

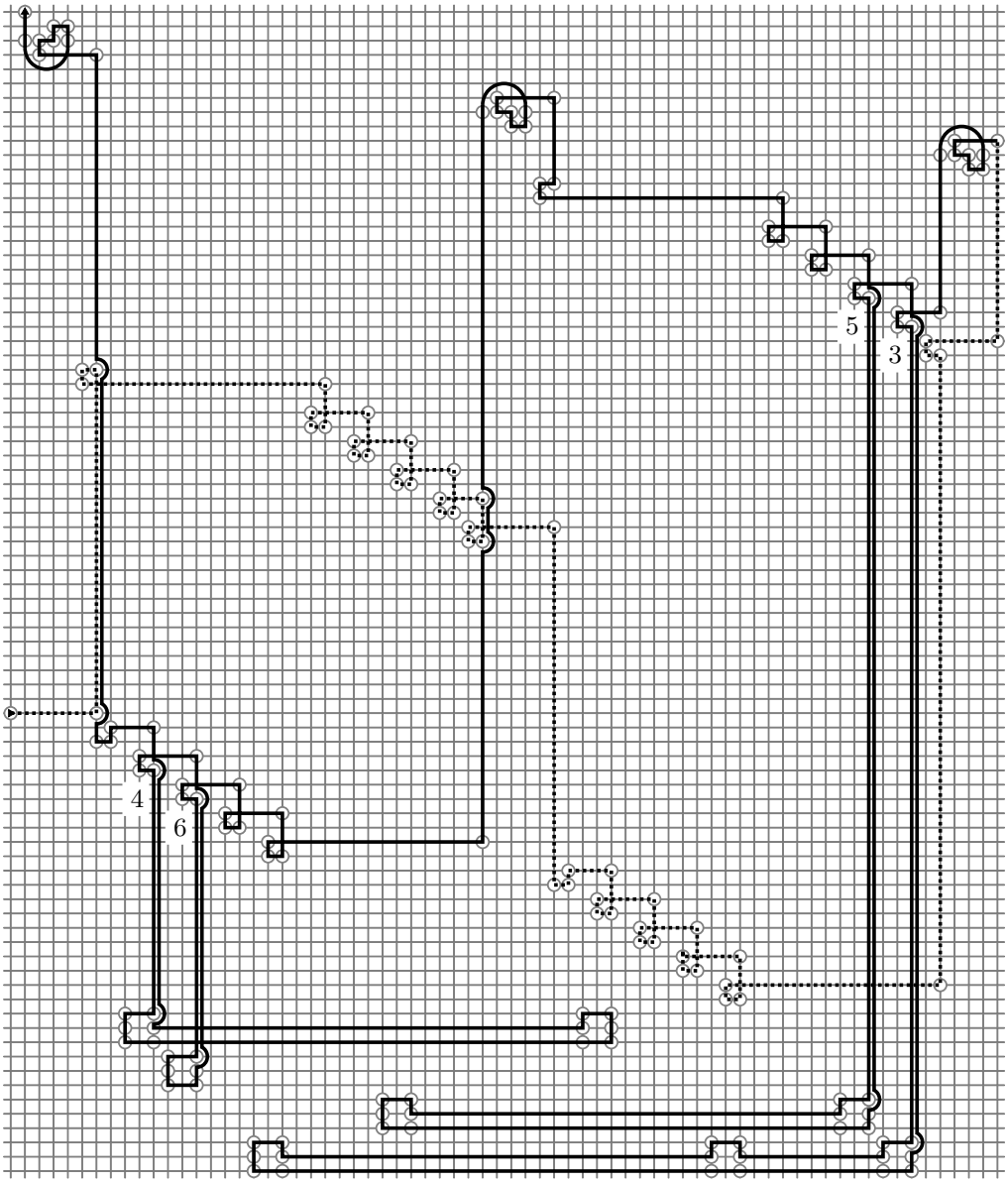


Figure 12: A solution to the example in Fig. 5. The dotted path shows the first stage $\mathsf{R}(t^*)$, with $t^*(x_1) = \top$ and $t^*(x_2) = \bot$. The solid path shows the second stage, with numbers indicating the alternative in Fig. 11 used to collect each clause.

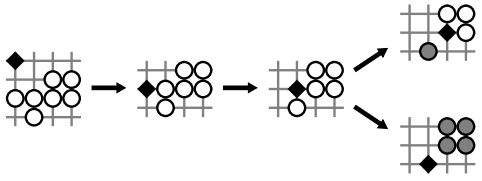


Figure 13: Starting at the topmost stone inevitably leads to a hopeless situation. A ◆ denotes the current position, and a ● denotes a stone in a dead end.

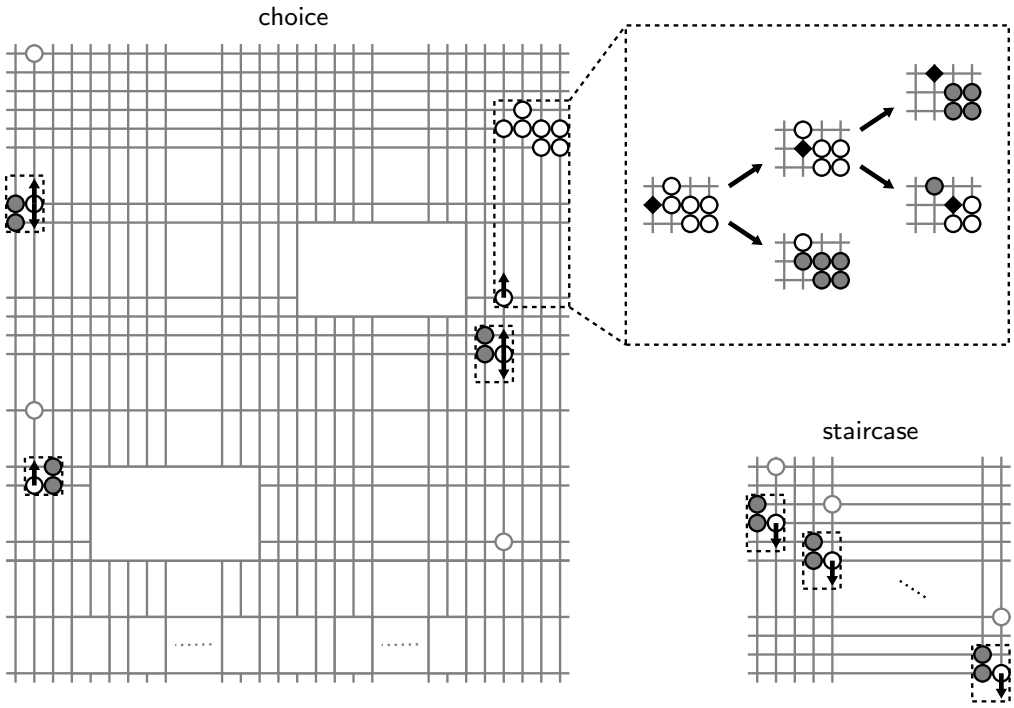


Figure 14: Possible deviations from the R-paths and the resulting dead ends.

We claim that there is an assignment t^* such that s starts with $\mathsf{R}(t^*)$. Figure 14 shows all the ways that one might attempt to deviate from the set of R-paths and the dead ends that would arise. By Lemma 1, we have that if this t^* were to fail to satisfy some clause C_k, then after $\mathsf{R}(t^*)$, the stones in the $\mathsf{c}(k, 1)$-components would together form a dead end. We conclude that the assignment t^* satisfies ϕ.

Acknowledgements. I thank Kristoffer Arnsfelt Hansen, who introduced me to Hiroimono and suggested the investigation of its complexity, and my advisor, Peter Bro Miltersen. I also thank the anonymous reviewers for their comments and suggestions.

References

[1] D. Andersson. Reduce 3-SAT to HIROIMONO. `http://purl.org/net/koda/s2h.php`.

[2] M. J. Costello. *The greatest puzzles of all time.* Prentice Hall Press, 1988.

[3] J. Culberson. Sokoban is PSPACE-complete. In *Proceedings of the International Conference on Fun with Algorithms*, pages 65–76. Carleton Scientific, 1998.

[4] E. D. Demaine, S. Hohenberger, and D. Liben-Nowell. Tetris is hard, even to approximate. In *Proceedings of the 9th Annual International Conference on*

Computing and Combinatorics, volume 2697 of *Lecture Notes in Computer Science*, pages 351–363. Springer-Verlag, 2003.

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

[6] R. Kaye. Minesweeper is NP-complete. *Mathematical Intelligencer*, 22:9–15, 2000.

[7] T. Yato and T. Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 86:1052–1060, 2003.

Paper 5

Hashiwokakero is NP-Complete

Daniel Andersson

Abstract. In a Hashiwokakero puzzle, one must build bridges to connect a set of islands. We show that deciding the solvability of such puzzles is NP-complete.

This paper is to appear in *Information Processing Letters*.

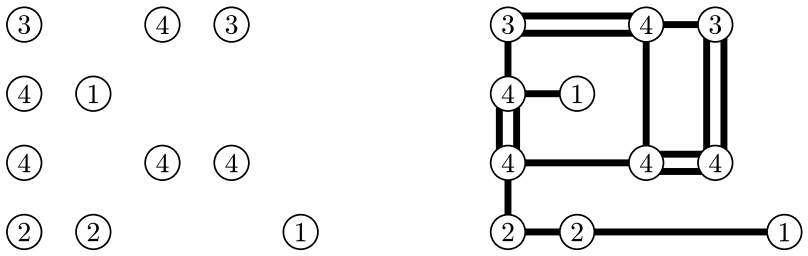


Figure 1: A Hashiwokakero puzzle and its solution.

1 Introduction

Hashiwokakero ("build bridges!") is a type of puzzle published by Nikoli [3]. Given a set of points in the plane (*islands*), one must connect them using vertical and horizontal line segments (*bridges*). Each island specifies how many bridges that must be connected to it. A bridge must not cross any other bridge or island, at most two bridges may run in parallel, and all islands must be reachable from one another. Fig. 1 shows an example.

Demaine and Hearn [1] survey complexity results on combinatorial games and puzzles, and they list Hashiwokakero as previously unstudied. We show:

Theorem 1. *Deciding the solvability of a Hashiwokakero puzzle is NP-complete.*

We construct a reduction from the following NP-hard problem [2]: Given a finite set $P \subset \mathbf{Z}^2$, is there a Hamiltonian circuit in the unit distance graph of P?

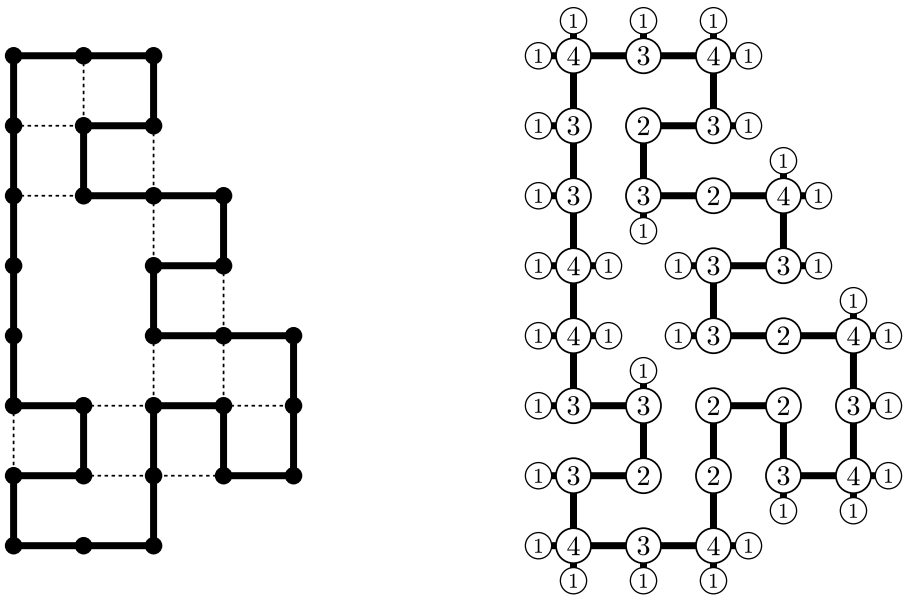


Figure 2: An example of the reduction and a pair of corresponding solutions.

2 Reduction

We assume $|P| \geq 3$. For each $x \in P$, output an island at $2x$ with bridge requirement $6 - |\{y \in P : |x - y| = 1\}|$ (*big island*) and an island at $x + y$ with bridge requirement 1 (*small island*) for each $y \in \mathbf{Z}^2 \setminus P$ such that $|x - y| = 1$. Fig. 2 shows an example.

Because of the reachability constraint, each small island must be connected to a big island, and by construction, there is only one choice. After all such connections have been made, all big islands have a remaining bridge requirement of 2. Since $|P| \geq 3$, the reachability constraint prevents parallel bridges between big islands, and so the connection possibilities now correspond exactly to edges in the unit distance graph of P.

References

[1] E. D. Demaine and R. A. Hearn. Playing Games with Algorithms: Algorithmic Combinatorial Game Theory. arXiv:cs/0106019v2 (earlier in MFCS 2001, LNCS 2136).

[2] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing* 11(4):676–686, 1982.

[3] Nikoli Co. Ltd. *Puzzle Cyclopedia.* 2004. ISBN 4-89072-406-0.

Paper 6

General Path Interdiction with Local Budgets

Daniel Andersson

Abstract. Khachiyan et al. considered network interdiction where the user finds a minimum-weight path after the interdictor removes edges subject to vertex-local budgets. They showed that optimal removals can be efficiently computed with a modified Dijkstra's algorithm. We extend their model by allowing partial interdiction of edges and using a more general definition of path weight, and we show that efficient algorithms still exist.

We also give an almost-linear time algorithm for the special case of bottleneck shortest path interdiction with vertex-local budgets of edge removals.

1 Introduction

Network interdiction is a classic type of problem in operations research, originally motivated by military applications [5], where the objective of Interdictor is to inhibit the usefulness of a network to User. Models that strive for realism are often solved as mixed-integer programs using advanced heuristics [3]. By contrast, in this paper, we focus on simple models and worst-case efficient algorithms.

Khachiyan et al. [4] considered a *shortest path* interdiction problem—a two-stage model on a weighted directed graph where Interdictor first removes some edges and User then finds a minimum-weight s-t path. Interdictor is constrained by some *budget*. When this budget is *global*—Interdictor may remove any k edges—finding the optimal choice is NP-hard [6]. Instead, Khachiyan et al. considered *vertex-local* budgets, where Interdictor may remove $k(v)$ outgoing edges from each vertex v, and showed this version to be solvable in polynomial time.

Khachiyan et al. also extended their results to a more general type of budgets, where each vertex v has a family $\mathcal{B}(v)$ of "affordable" subsets of $\text{out}(v)$ (the outgoing edges from v). Naturally, $\mathcal{B}(v)$ is assumed to be downward closed with respect to set inclusion (also known as an *independence system*).

In this paper, we further generalize the model of [4] by adding the capability of *partial interdiction*—instead of just removing edges, Interdictor increases their weights. We also generalize how User computes the weight of a path. Last, we consider the special case of *bottleneck interdiction*, where User finds a path minimizing the *maximum* weight. We show that for vertex-wise budgets of edge removals, this case can be solved in almost-linear time.

2 General Path Interdiction

2.1 Definitions

We are given a simple directed graph $G = (V, E)$ with a source s and a sink t reachable from any vertex. Interdictor will first assign a non-negative weight to each edge, and User will then find a minimum-weight s-t path.

At each vertex u, Interdictor selects a function $w_u : \text{out}(u) \to \mathbf{R}_+$ from a given budget $W_u \subseteq [\text{out}(u) \to \mathbf{R}_+]$. We require that the budget, when viewed as a subset of $\mathbf{R}_+^{|\text{out}(u)|}$, is non-empty, closed in the topological sense, and downward closed with respect to the weak component-wise order.

Let $w := \bigcup_{u \in V} w_u$. Extend w to paths by defining $w(\epsilon) := 0$ and $w(e\pi) := p(w(e), w(\pi))$ for any edge e and path π, where ϵ denotes the empty path and $p : \mathbf{R}_+^2 \to \mathbf{R}_+$ is a given operator that is continuous and non-decreasing in each argument and such that $p(0, x) \geq x$ for all $x \in \mathbf{R}_+$. User finds an s-t path π^* minimizing w.

General path interdiction with local budgets is the computational problem, given G, s, t, p and budgets W_u for each vertex u, to find the supremum x^* of all $x \in \mathbf{R}_+$ such that Interdictor can ensure that $w(\pi^*) \geq x$. Note that x^* may be ∞. We call x^* the *value* of the instance, and we define the value of a vertex u as the value of the modified instance where $s := u$.

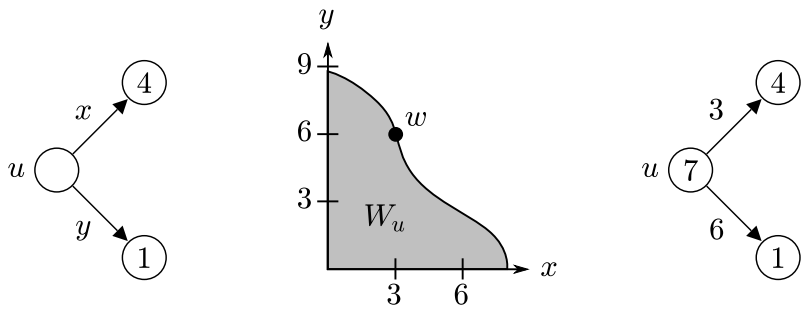


Figure 1: An example showing the task of L. Here, p is addition. An optimal choice of w within the budget W_u (*middle*) ensures that both $w(x) + 4$ and $w(y) + 1$ are at least 7 (*right*).

We can reduce the model of Khachiyan et al. to ours as follows: Let p be the addition operator. Given a global weight function $w : E \to \mathbf{R}_+$ and for each vertex u a family $\mathcal{B}(u)$ of affordable subsets of $\mathrm{out}(u)$, define our budget for each u as $W_u := \cup_{X \in \mathcal{B}(u)} \{z : \mathrm{out}(u) \to \mathbf{R}_+ | \forall e \in \mathrm{out}(u) \setminus X : z(e) \leq w(e)\}$.

2.2 Algorithm

Below, we give an algorithm for solving general path interdiction with local budgets. We denote by Ev the set of vertices with edges to v, and symmetrically for vE. We assume that the operator p and the budgets W_u are given in the form of an oracle L that can solve the following "local" optimization problem: Given u, a subset $S \subseteq uE$, and a mapping $d : S \to \mathbf{R}_+ \cup \{\infty\}$, find the supremum of all $z \in \mathbf{R}_+$ such that there is $w \in W_u$ with $p(w(u, v), d(v)) \geq z$ for all $v \in S$. (We extend $p(x, \infty) := \infty$ for all x.) See Figure 1 for an illustration. As another example, for polyhedral budgets and linear p, the oracle L can be implemented using linear programming.

Algorithm *GPI*
1. $d[V] \leftarrow \infty$
2. $d[t] \leftarrow 0$
3. $Q \leftarrow V$
4. **while** $Q \neq \emptyset$
5. **do** $v \leftarrow \arg\min_{u \in Q} d[u]$
6. $Q \leftarrow Q \setminus \{v\}$
7. **for** $u \in Ev \cap Q$
8. **do** $d[u] \leftarrow L(u, uE \setminus Q, d)$
9. **return** $d[s]$

The mapping d contains an upper bound on the value of each vertex, and when a vertex u is removed from Q, then $d[u]$ is exactly the value of u. The full correctness proof is analogous to that of [4, Section 2.3], hence omitted. Using a Fibonacci heap to store the elements of Q ordered by their values in d, the running time is $O(|E| + |V| \log |V|)$ with $O(|E|)$ calls to the oracle L.

3 Bottleneck Shortest Path Interdiction

3.1 Algorithm

We now consider the special case where p is the maximum operator (i.e., User finds the *bottleneck*[1] shortest path), there is a global weight function $w : E \to \mathbf{R}_+$ and each vertex u has a budget of $k(u)$ edge removals. The following algorithm is similar to the "modified algorithm" in [4, Section 2.4].

Algorithm *BSPI*

```
1.  for u ∈ V do b[u] ← k(u)
2.  for e ∈ E do c[e] ← w(e)
3.  T ← {t} (* finished vertices *)
4.  P ← in(t) (* edges across the cut (V \ T, T) *)
5.  while P ≠ ∅
6.      do (u, v) ← arg min_{e∈P} c[e]
7.         P ← P \ {(u, v)}
8.         if b[u] > 0 (* greedily remove (u, v) if possible *)
9.           then E ← E \ {(u, v)}
10.                b[u] ← b[u] − 1
11.          else T ← T ∪ {u}
12.               if u = s then return c[u, v]
13.               for e ∈ in(u) \ T (* propagate the value of u *)
14.                   do c[e] ← max{c[e], c[u, v]}
15.                      P ← P ∪ {e}
16. return ∞
```

Using a binary heap priority queue for the elements of P gives an $O(|E| \log |V|)$ running time. However, we can obtain an almost-linear running time by using the following technique that Gabow and Tarjan [2] successfully applied to other bottleneck graph problems.

First, note that the universe of keys in the priority queue is $w(E)$. Thus, we may first sort all weights and replace them by integers $1, \ldots, |E|$. Then, since extractions are monotone, all our priority queue operations can be implemented in amortized constant time by keeping an array of $|E|$ buckets. Thus, when weights are already sorted, *BSPI* runs in $O(|E|)$ time.

The remaining challenge is to avoid completely sorting the weights. For example, if weights are first partitioned into "light" and "heavy", and *BSPI* is then run on this "coarse" problem (having only two edge weights), it is not hard to see that it will correctly classify the value of the instance as either light or heavy. If the value is light, then all heavy weights can be combined into one, and *vice versa*. In any case, the number of weights is (roughly) halved. A more sophisticated coarsening scheme (see [1, 2]) yields a total running time of $O(|E| \log^* |V|)$. This matches the best known bound without interdiction—we get interdiction "for free".

[1] The term "bottleneck" is more apt if weights are viewed as inverted capacities.

3.2 Remarks

There is a linear-time reduction from solving Washburn's *deterministic graphical games* [1] to our bottleneck interdiction problem. The basic idea, in the terminology of [1], is to let $k(v) := 0$ for vertices belonging to player Min and $k(v) := |\text{out}(v)| - 1$ for vertices belonging to player Max. Thus, a linear-time algorithm for bottleneck interdiction would settle the remaining open problem of [1].

Finally, we note that the *global* budget version of bottleneck interdiction (remove any k edges) is solvable in polynomial time, by performing a binary search for the maximum weight x such that removing all edges with weight at least x yields a network with s-t edge connectivity at most k.

Acknowledgements. The author thanks Anastasios Viglas for inspiring discussions and acknowledges partial support from the Center for Algorithmic Game Theory funded by the Carlsberg Foundation.

References

[1] D. Andersson, K. A. Hansen, P. B. Miltersen, and T. B. Sorensen. Deterministic graphical games revisited. In *Proceedings of the Fourth Conference on Computability in Europe (CiE 2008)*, volume 5028 of *LNCS*, pages 1–10. Springer, 2008.

[2] H. Gabow and R. Tarjan. Algorithms for two bottleneck optimization problems. *J. Algorithms*, 9:411–417, 1988.

[3] E. Israely and K. Wood. Shortest-path network interdiction. *Networks*, 40:97–111, 2002.

[4] L. Khachiyan, E. Boros, K. Borys, K. M. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233, 2008.

[5] A. McMasters and T. Mustin. Optimal interdiction of a supply network. *Naval Research Logistics Quarterly*, 17:261–268, 1970.

[6] B. Schieber, A. Bar-Noy, and S. Khuller. The complexity of finding most vital arcs and nodes. Technical Report UMIACS-TR-95-96, University of Maryland Institute for Advanced Computer Studies, College Park, MD, USA, 1995.