CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Computer Science

Master's Thesis

Computing Stackelberg Equilibrium with Memory in Sequential Games

Michael Hlaváček

May 2018

Supervisor: Mgr. Branislav Bošanský, Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hlaváček** Jméno: **Michael** Osobní číslo: **425079**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Výpočet Stackelbergových strategií s pamětí v sekvenčních hrách

Název diplomové práce anglicky:

Computing Stackelberg equilibrium with Memory in Sequential Games

Pokyny pro vypracování:

Stackelberg Equilibrium (SE) is a solution concept where the leader commits to a strategy that is observed by the follower that plays a best response. There are several known theoretic and algorithmic results for computing Stackelberg equilibria in sequential games. However, the existing works focus on behavioral strategies where the player chooses from a probability distribution over actions in each state of the game. In compactly represented games (e.g., finite games played represented as DAGs, infinite stochastic games), committing to behavioral strategy is not necessarily optimal. The goal of the student is to (1) analyze the required memory for finding Stackelberg equilibrium for finite and infinite games, (2) analyze the computational complexity for computing this equilibrium, (3) design and implement an algorithm for computing SE with memory for selected classes of games, and (4) experimentally analyze the scalability of the new algorithm.

Seznam doporučené literatury:

[1] B. Bosansky, J. Cermak; Sequence-Form Algorithm for Computing Stackelberg Equilibria in Extensive-Form Games, AAAI 2015
[2] B. Bosansky, S. Branzei, K. A. Hansen, P. B. Miltersen, T. B. Lund; Computation of Stackelberg Equilibria of Finite Sequential Games, ACM TEAC 2017
[3] J. Cermak, B. Bosansky, K. Durkota, V. Lisy, C. Kiekintveld; Using Correlated Strategies for Computing Stackelberg Equilibria in Extensive-Form Games, AAAI 2016
Computing Optimal Strategies with Memory to Commit to in Sequential Games

Jméno a pracoviště vedoucí(ho) diplomové práce:

Mgr. Branislav Bošanský, Ph.D., centrum umělé inteligence FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **09.02.2018** Termín odevzdání diplomové práce: **25.05.2018**

Platnost zadání diplomové práce: **30.09.2019**

Mgr. Branislav Bošanský, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. .....
Datum převzetí zadání

.....
Podpis studenta

Acknowledgement / Declaration

I would like to thank my supervisor Branislav Bošanský for his patience, insights, and support. Also, I would like to thank my family for their patience and support.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 25.5.2018

I declare that the presented work was developed independetly and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university thesis.

Prague, date 25.05.2018

Abstrakt / Abstract

Tato práce se zabývá problémem representace Stackelbergových strategii v kompaktně representovaných sekvenčních hrách pomocí strategií s pamětí, a hledáním takových strategií.

Dokazujeme že pro hry, které jsou reprezentované orientovaným acyklickým grafem bez náhodných stavů, stejně jako pro stochastické hry na grafech bez náhodných stavů, postačuje lineární množství pamětových stavů. Pro případ orientovaných acyklických grafů předkládáme polynomiální algoritmus pro hledání Stackelbergových strategií s pamětí.

Pro hry na orientovaném acyklickém grafu s náhodnými stavy existuje kvadratická mez na potřebné paměťové stavy, avšak nalezení takové strategie je NP-těžké. Prezentujeme aditivní aproximační algoritmus pro tuto třídu her a experimentálně jej testujeme na náhodně generovaných hrách.

Klíčová slova: teorie her, sekvenční hry, Stackelbergova rovnováha, strategie s pamětí

Překlad titulu: Výpočet Stackelbergových strategií s pamětí v sekvenčních hrách

In this thesis we consider the problem of representing Stackelberg equilibria (sometimes called leader-follower equilibria) in compactly represented sequental games as strategies with memory, and the problem of finding such equilibria.

We show that in games represented by a directed acyclic graph without chance nodes and in stochastic games without chance nodes, linear number of memory states suffice. We provide polynomial time algorithm for the DAG case.

In games represented by a DAG with chance nodes, we give a quadratic bound on used memory states. We show that the problem of finding such strategies is NP-hard. We provide an additive aproximation algorithm for such games, and experimentally evaluate this algorithm on randomly generated games.

Keywords: game theory, sequential games, Stackelberg equilibrium, strategies with memory

/ Contents

Chapter 1
Introduction

Imagine you are walking on a sidewalk, looking to the ground. When you raise your eyes, you chatch the eysight of a stranger coming towards you. Both of you know that if you continue walking like you are, you will clash. Both of you decide which way to sidestep. Both of you sidestep in the same direction. You try to step aside once again, and once again both of you sidestep in the same direction. After many more sidesteps, and a somewhat awkward smile, you manage to dodge each other. Such situations, and many, many more, are studied in the field of game theory.

The field of game theory studies situations in which multiple actors act to maximize their own profit. While the term game theory originated from considering game in the everyday sense of the word, as chess or rock-paper-scissors, it encompasses much richer kinds of situations. The players of a game can be two companies selling shoes, the ticket inspectors and passangers who do not want to pay, or security officers at the airport and smugglers of rare species of birds.

Many different formalizations of such situations are used. Ever since the establishment of the field, a distiction is made between normal-form games, in which all players act simultaneously, and extensive-form games, in which the players take turns. Games may include uncertainty, either in outcomes of some actions, in which case a special player often called nature randomizes over outcomes, or in information about the state of the game, in which case the game is said to be of imperfect information. A classic example of a game with imperfect information and uncertainty is the game of poker – no player knows the hand of other players, and the cards are dealt from the deck randomly.

When considering solutions of these formalizations, many different concept arise and are used. The traditional solution is a Nash equilibrium, which is a strategy in which no player wants to single-handedly play another strategy. Another solution is a Stackelberg equilibrium, in which case there is a special player called leader who has the ability to decide beforehand how he wants to play. When the Stackelberg equilibrium is used, other players must believe that the leader will actually follow his commitment. In practice, this can be because the leader is a big company with a monopoly (which was the original motivation for this solution concept), or perhaps because the leader is a security protocol of a state, which has to be decided beforehand. In this thesis, Stackelberg equilibria are considered.

While game theory began as a theoretical subject, with the invention of computers the problem of computing solutions arose. In order to be computed effectively, one has to input a description of the problem which is compact enough. While there is already an exponential step between normal-form games and extensive-form games, even the extensive-form games are in practice too huge to be computed. For example, a variant of poker 2 player limit Texas Hold'em has 10^{17} states. Therefore, there is a need for event more compact representations. One way to reduce the size of extensive-form games is to consider directed acyclic graphs (DAGs) instead of trees. As an example, consider a game of chess. In the traditional extensive-form game description, there is

one state for every possible way a certain configuration of the board can be reached. In the case of DAGs, there can be just one game state for every configuration of the board.

However, it may happen that by considering DAGs instead of game trees, we lose the ability to efficiently compute a solution if we require just one strategy for each game state, or that this solution is worse than that of the game on a tree. To overcome this hurdle, a compromise is used in the form of strategies with memory. In such strategies, the players are allowed to remember some information about the past progress of the game, but ideally not the complete history as it is in extensive-form games. In this thesis, we show that there are bounds on the size of the memory needed to represent optimal strategies, and that for some games it can be efficiently found.

1.1 Related work

The solution concept of Stackelberg equilibria was first introduced in [14], when reasoning about an economic problem of duopoly. The existence of Stackelberg equilibria in mixed strategies was studied by von Stengel and Zamir [15], who also showed that in a two-player normal-form game, commiting to a strategy can never hurt a player when compared to a Nash equilibrium. The problem of computing Nash equilibria was first considered by Conitzer and Sandholm [2]. They show that for a normal-form game, a Stackelberg equilbrium in pure strategies can be found in polynomial time for any number of players. They provide a polynomial algorithm for computing Stackelberg equilibria in mixed strategies for two players, but show that for three players the problem of finding such an equilibrium is NP-hard. The same authors considered Stackelberg strategies in extensive-form games [8]. They show that almost all problems of finding Stackelberg equilibria in extensive-form games are NP-hard, with some notable exceptions. They show that the problem is polynomial for the cases of two-player games without chance nodes on a tree when considering both mixed and pure strategies, and for the case of a game on a DAG without chance nodes when considering pure strategies. Later, the same authors focused on commitement in stochastic games [9]. Bošanský et al [1] considered commitment to correlated strategies, and provided an approximation algorithms for two player games on a tree with chance nodes.

Another line of work is on the subject of stochastic games. Notably, Gupta et al.[4] considered Stackelberg equilibria with memory in discounted sum games. Discounted sum games are played on directed graphs without sinks, with utilities assigned to edges, and each turn are discounted by some discount factor λ. The players try to maximize discounted sum of their utilities. The authors show that in such games, sometimes an infinite memory is needed to represent pure-strategy Stackelberg equilibria. Later, Gupta [3] considered Stackelberg equilibria in mean payoff games. They are played on the same structure as discounted sum games, but the goal is different - the players try to reach average utility. As for the real-world applications of not only Stackelberg equilibria, Tambe [13] provides an overview of applications of game theory to the domain of security.

1.2 Outline of this thesis.

In this thesis, we address the problem of compact representation of Stackelberg equilibria on compactly represented games. In the second chapter, we provide the necessary

theoretical background to game theory. We introduce compact game representations – games represented by directed acyclic graphs, or even graphs with cycles.

In the third chapter, we present our main results. For games without chance nodes on a directed acyclic graph, we show that for every Stackelberg equilibrium there is a strategy with memory which uses no more than $|\mathcal{S}_2| + |\mathcal{Z}| + 1$ states. We present an algorithm that finds this strategy in $O(|\mathcal{S}_2|^2 |\mathcal{S}|^3)$. Then, we examine stochastic games without chance nodes on general graphs. We show that the memory bound $|\mathcal{S}_2|+|\mathcal{Z}|+1$ still holds, and show that in fact every Stackelberg equilibrium is finite. Finally, we examine games on directed acyclic graphs with chance nodes. We show that the memory bound becomes quadratic, namely $|\mathcal{S}_1|\,|\mathcal{S}_2| + 1$. We show that finding optimal strategy with memory is NP-hard. Finally, we present an approximation algorithm which find a strategy for which the expected utility is withing ϵ of the optimal one.

In the last chapter, we experimentally evaluate our implementetion of ϵ aproximation algorithm for games on directed acyclic graphs on randomly generated games. We confirm that the runtime of the algorithm indeed behaves according to our theoretical guarantees. We examine how many memory states are actually needed, and find that the bound on memory states is rather loose in practice.

Chapter 2
Game Theory

Game Theory deals with rational decisions when multiple decision makers are involved. Despite it has a somewhat misleading name, it does not only concern games in the everyday use of that word, but a much broader class of problems which concern multiple actors. These actors can either cooperate to achieve a good collective outcome, which is the subject of cooperative game theory, or to selfishly maximize their own profit, which is the domain of competitive game theory. In this chapter, we introduce the concepts used in this field.

2.1 Normal-form games

Normal form games are the simplest class of games studied. They represent one-shot games, where all players make a single decision how to act, and are rewarded according to the combination of their actions. Although this thesis deals with games in extensive form, which will be introduced later, many of the concepts used to describe and solve normal-form games translate to games in extensive form, and are much more easily explained on games in normal form.

Definition 2.1. [12] A *normal-form game* is a tuple (N, A, u), where

- $N = \{1, \dots n\}$ is a finite set of players,
- $A = A_1 \times \dots \times A_n$, where A_i is a finite set of actions available to player i. A vector $a = (a_1, \dots, a_n) \in A$ is called an *action profile* and
- $u = (u_1, \dots, u_n)$, where $u_i : A \to \mathbb{R}$ is a real-valued *utility function* of player i.

		Player 2		
		Rock	Paper	Scissors
Player 1	Rock	$0, 0$	$-1, 1$	$1, -1$
	Paper	$1, -1$	$0, 0$	$-1, 1$
	Scissors	$-1, 1$	$1, -1$	$0, 0$

Figure 2.1. A normal-form representation of rock-paper-scissors. The labels of rows and columns represent actions of both players, the tuples in the cells of matrix represent utilities of both players.

In Figure 2.1 there is an example of a normal form game, a game of rock-paper-scissors, represented as a matrix. Rows represent actions of player 1, columns actions of player 2. The pairs of numbers in the cells represent utilities of both players. If a player wins, he is given utility 1, if he loses, he is given -1. If the game results in a draw, both players are awarded 0. A simpler game is shown in figure 2.2. In this game, two players are walking on a sidewalk towards each other, and try to avoid collision. If they both coose the same direction (they are facing opposite ways) to sidestep, they

		Player 2	
		Left	Right
Player 1	Left	$1, 1$	$0, 0$
	Right	$0, 0$	$1, 1$

Figure 2.2. A coordination game. Two people walk towards each other, trying to avoid collision. If they both choose the same direction to sidestep, they pass each other succesfully, receiving utility 1. Otherwise, they bump into each other, receiving lower utility. This game has two pure-strategy Nash equilibria, and illustrates the famous equilibrium selection problem.

succesfully dodge each other, receiving utility 1. Otherwise, they bump into each other, receiving utility 0.

These two examples lead us to the way normal form games are played. In the simplest form, both players choose one action to play. Such strategies are called *pure strategies*, and a collection of pure strategies of all players is called a *pure strategy profile*. Pure strategy profiles, however, do not provide a strong-enough solution concept. Consider a game of rock-paper-scissors. If your opponent knows what action you are planning to do, he can always choose a winning moves. Therefore, for every pure strategy profile we can find a player that can change his action to obtain better results. In formal terms, such a strategy profile is not *stable*. Therefore, we need to indtroduce stronger strategies - the players are allowed to choose an action according to some probability distribution.

Definition 2.2. [12] Let (N, A, u) be a normal-form game, and for any set X let $\Delta(X)$ be the set of all probability distributions over X. Then the set of *mixed strategies* of player i is $\Pi_i = \Delta(A_i)$, and a member σ of the cartesian product of strategy sets for each player $\Pi_1 \times \ldots \times \Pi_n$ is a *mixed strategy profile*.

To make reasoning about reasoning about strategy profiles easier, we can also define a strategy profile of all players except i as $\sigma{-}i = (\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_n)$, and write $\sigma = (\sigma_i, \sigma_{-i})$. This definition of strategies overcomes the problems with pure strategies. Note that pure strategy profiles now became a special case of mixed strategy profile – each player can choose to play a certain strategy with probability 1. To fully define the concept of mixed strategies, we have to define the utility of playing a certain strategy profile. Without surprise, we define it as the expected value of the utility obtained by playing the strategy profile.

Definition 2.3. [12] In a normal form game (N, A, u), the *expected utility* u_i of player i in a strategy profile $\sigma = (\sigma_1, \ldots, \sigma_n)$ is defined as

$$u_i(s) = \sum_{a \in A} u_i(a) \prod_{j=1}^{n} \sigma_j(a_j).$$

Using this definition of utility, we can now focus our attention to solving normal form games. To do so, we formalize the concept we discussed when reasoning about pure strategies – the motivation of individual players to switch a strategy from a predetermined one.

Definition 2.4. [12] A *best response* of player i to the strategy profile σ_i is a mixed strategy $\sigma_i^* \in \Pi_i$ such that $u_i(\sigma_i^*, \sigma_{-i}) \geq u_i(\sigma_i, \sigma_{-i})$ for all strategies $\sigma_i \in \Pi_i$. The set of all best responses to σ_{-i} is denoted as $\mathcal{BR}(\sigma_{-i})$.

In plain terms, a strategy s_i is a best response to s_{-i} if player i cannot obtain better utility by playing another action. Consider the coordination game in Figure 2.2, and

a strategy where player 1 plays Left and player 2 plays Right. Then the strategies are not best responses to each other - player 1 can obtain a better utility by playing Right, and player 2 can obtain a better utility by playing Left. Using the definition of best response we can now define the most famous solution concept in game theory, the Nash equilibrium.

Definition 2.5. [12] A strategy profile s is a *Nash equilibrium* if for all agents i for all players i it holds that s_i is a best response to s_{-i}.

As an example, consider the coordination game in Figure 2.2. We can see two Nash equilibria, one when both players play Right, and one when both players play Left. On the other hand, the game of rock-paper-scissors in Figure 2.1 does not have a pure strategy Nash equilibrium – each player can always best-respond by plaing the winning move. The equilibrium in mixed strategies exists – both players can play each action with probability $\frac{1}{3}$. This leads us to the question of existence of a Nash equilibrium, which was famously answered by John Nash (hence the name of the equilibrium).

Theorem 2.1. [10] Every game with a finite number of players and action profile has at least one Nash equilibrium.

As we saw on the example of the coordination game, a game may have multiple Nash equilibria. This leads to an interesting problem. Imagine two opponents playing the coordination game. Each one of them computes a Nash equilibrium (as they both know game theory), but player 1 finds an equilibrium $(Left, Left)$ and player 2 finds $(Right, Right)$. When playing according to these computed equilibria, they both receive the utility 0. This is called an *equilibrium selection problem*, and is widely studied as it is important when applications of game theory are considered (see for example [5]).

Another concept of solution adresses, among other things, this problem. Consider a situation in which one player, called the leader, has the ability to anounce his strategy beforehand, and the other players, called followers, play a Nash equilbrium with respect to the leader's strategy. This solution concept is called a Stackelberg equilibrium, as it was first defined by von Stackelberg [14].

Definition 2.6. [3] A *Strong Stackelberg equilibrium* in a normal-form game (N, A, u) is a strategy profile $\sigma = (\sigma_1, \ldots, \sigma_n)$ such that

$$\sigma = \underset{\sigma'_1 \in \Pi_1, \sigma'_i \in \mathcal{BR}(\sigma'_{-i})}{\operatorname{argmax}} u_1(\sigma)$$

While it may seem that by the requirement to commit to the strategy before the game is played and letting the other players exploit the knowledge hurts the leader, the opposite is actually true. Indeed, the leader can always commit to playing a Nash equilibrium. In the real world applications, one has to ensure that the leader can actually credibly commit to playing a certain strategy. This happens for example if the actors in the game are companies, one of which is much bigger (this was the original motivation of the definition of Stackelberg equilibrium).

While equilibria provide good solutions when considering self-interested players, the give the player no guarantee should his/her opponents choose to harm them. However, there is another solution concept which provides a guaranteed minimal outcome.

Definition 2.7. [12] The *maxmin strategy* for player i is

$$\underset{\sigma_i \in \Pi_i}{\operatorname{argmax}} \min_{\sigma_{-i} \in \Pi_{-i}} u_i(\sigma_i, \sigma_{-i})$$

The *maxmin value* of player i is

$$\mu_i = \max_{\sigma_i \in \Pi_i} \min_{\sigma_{-i} \in \Pi_{-i}} u_i(\sigma_i, \sigma_{-i})$$

This definition assumes the worst - that the other players play to collectively harm i, and then maximizes the outcome. As such, when playing his maxmin strategy the player is always guaranteed at least his maxmin value (as deviation by another player can ony increse his utility). We now turn our attention to sequential games.

2.2 Sequential games

Many of the problems solved by game theory have a sequential nature. Consider for example a game of chess, in which both players take turns, moving one figure at a time. While it is possible to represent such a game in normal form, it is not practical. One would need one action for every possible sequence of moves from start of the game to the end. To solve this, we turn to another model, where the games are represented as decision trees, or (more-generally), graphs. Because this thesis deals with two-player games only, we will state the definitions only for such games. When talking about Stackelberg equilibria, we will call player 1 the *leader* and player 2 the *follower*.

Definition 2.8. A *two-player sequential game* is a tuple $G = (\mathcal{N}, \mathcal{S}, \mathcal{Z}, \rho, \mathcal{A}, u, \mathcal{T}, \sigma_0, s_0)$ where

- $\mathcal{N} = \{1, 2\}$ is a set of players,
- $\mathcal{S}$ is a set of non-terminal states (also called nodes),
- $\mathcal{Z}$ is a set of terminal states (also called leaves)
- $\rho : \mathcal{S} \to \mathcal{N} \cup \{0\}$ is a function which defines which player plays in s, or whether the node is a chance node ($\rho(s) = 0$); we denote $S_i = \{s \in S | \rho(s) = i\}$ the set of all states in which player i plays,
- $\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}(s)$ is the set of actions, where $\mathcal{A}(s)$ is the set of actions available in a state s; we denote $\mathcal{A}_i$ the set of actions available to player $i \in \mathcal{N} \cup \{0\}$,
- $u = (u_1, u_2)$ where $u_i : \mathcal{Z} \to \mathbb{R}$ is the utility function of player $i \in \mathcal{N}$,
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S} \cup \mathcal{Z}$ is a transition function which determines what state follows when player $\rho(s)$ plays action $a \in \mathcal{A}(s)$ in state $s \in \mathcal{S}$,
- $\sigma_0 : \mathcal{S}_0 \times \mathcal{A}_0 \to [0, 1]$ are the probabilities of actions in every node such that $\rho(s) = 0$, such that $\sum_{a \in \mathcal{A}_0(s)} \sigma_0(s, a) = 1$ and
- s_0 is the initial state.

Because we are interested in playing sequences of actions starting in s_0, we assume that in the graph $(\mathcal{S}, \{(s_1, \mathcal{T}(s_1, a) | a \in \mathcal{A}(s)\})$ every state is reachable from s_0. We will call a *history* (or *path*) a sequence $\pi = v_0 a_0 \dots v_{k-1} a_{k-1} v_k$ such that $v_0 = s_0$, for all $0 \leq i \leq k-1$ it holds that $v_i \in \mathcal{S}$, $v_k \in \mathcal{S} \cup \mathcal{Z}$ and for all $0 \leq i \leq k-1$ it holds that $\mathcal{T}(v_i, a_i) = a_{i+1}$. That is, a history is a valid path through the game. Additionaly we let $\pi[i] = v_0 a_0 \dots v_i$ be a prefix of π up to the node v_i (note that the action a_i is not included), and $\pi[i{:}j] = v_i a_i \dots v_j$ a part of π between indexes i and j. Finally, let πa be a history obtained by playing action a after π, that is $\pi a = v_0 a_0 \dots v_k a \mathcal{T}(v_k, a)$.

The definition of a two-player sequential game is quite general, and engulfs several classes of games studied in game theory. We will split the discussion into three sections, depending on the type of graph induced by the transition function $\mathcal{T}$. It can be a tree, (in which case the game is called a *game in extensive form*), a directed acyclic graph or a general graph (in which case the game is called a *stochastic game*).

2.2.1 Extensive-form games

Extensive form games are sequential games in which the graph $G = (\mathcal{S}, \{(s_1, \mathcal{T}(s_1, a) | a \in \mathcal{A}(s)\})$ is a tree. An example is given in Figure 2.3. It is a modification of the coordination

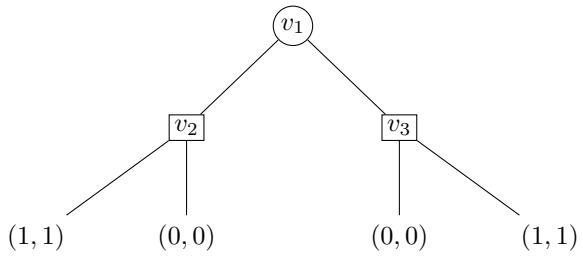

Figure 2.3. An example of a game in extensive form. It is an analogue of the coordination game from Figure 2.2, with the modification that player 1 is allowed to move first.

game from Figure 2.2. Player one, whose states are depicted as circles, plays first, deciding whether to move Left or Right. Then, player two, who controlls the square nodes, moves, facing the same decision.

Because the transition fuction of extensive-form games defines a tree, for each state there is a unique way it can be reached. Therefore, the strategy of a player in such a game only needs to assign an action to each state the player controlls (as opposed to assigning a strategy to every path from the root node, as we will do with games on more complex graphs).

Definition 2.9. [12] Let G be an extensive-form game. A pure strategy σ_i of player i is a member of the Cartesian product $\prod_{s \in S_i} \mathcal{A}(s)$.

Note that this definition of strategy requires us to choose an action for *every* node in which the player plays, regardles of whether this node can be reached when following this strategy. This property is actually useful - it allows us to create a normal-form game by letting the actions of player i in the normal form games be all pure strategies in the original extensive form game. For example, the extensive form game in Figure 2.3 corresponds to the normal-form coordination game in Figure 2.2. We now turn our attention to randomized strategies. There are two options in which these strategies can be formalized. One option is to assign probability to every pure strategy (as we did with normal-form game). The second option is to assign a probability distribution to each node of the tree. This motivates the two following definitions

Definition 2.10. [11] A *mixed strategy* μ_i of player i in an extensive-form game G is a probability measure over the set of player i's pure strategies. A *strategy profile in mixed strategies* $\mu = (\mu_1, \mu_2)$ is a collection of strategies for both players.

Definition 2.11. [11] A *behavioral strategy* σ_i of player i is a collection $(\sigma_i(s, \cdot))_{s \in \mathcal{S}_i}$ of independent probability measures, where $\sigma_i(s, \cdot)$ is a probability measure over $\mathcal{A}(s)$, with $\sigma_i(s, a)$ being the probability of playing action a in s.

Once again, we call Π_i the set of all behavioral strategies of player i. To make our lives easier, we will drop the index i when speaking about behavioral strategies, as the player whose turn it is is uniquely determined by the state, that is $\sigma(s, a) = \sigma_{\rho(s)}(s, a)$. Because this thesis concerns only games with perfect information (that is games in which both players always know the state of the game, for formal definition we encourage the reader to see [12]), due to Kuhn [6] we know that the mixed and behavioral strategies are in fact equivalent. Therefore, we will proceed with only behavioral strategies in mind.

Because extensive form games are played on a tree, for each state $n \in \mathcal{S} \cup \mathcal{Z}$ there is a unique history leading to it, that is there is a unique $\pi = v_0 a_0 \dots v_{k-1} a_{k-1} n$. To express the probability of reaching n when playing according to a strategy profile σ, we

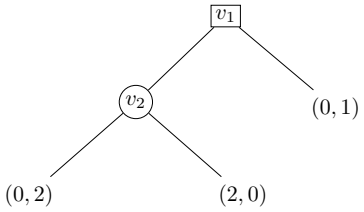

Figure 2.4. An example of an extensive form game to illustrate the properties of Stackelberg equilibrium.

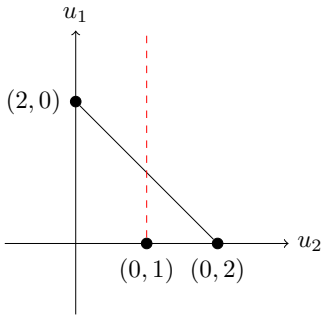

Figure 2.5. The visualisation of possible outcomes of the game in Figure 2.4 in the space of utilities. The red dashed line signifies the maxmin value $\mu(v_1)$ of the follower's node v_1. It splits the space into two half spaces, where the half space to the left of this line is unreachable at v_1 - the follower can always play right.

can use this unique π and define $p(n) = \prod_{i=0}^{k-1} \sigma(v_i, a_i)$. Using this, we can express the expected utility of a strategy.

Definition 2.12. [11] An *expected utility* of a behavioral strategy profile σ for player i is $u_i(\sigma) = \prod_{z \in \mathcal{Z}} p(z) u_i(z)$.

Using this definition, we can define the best response to a strategy.

Definition 2.13. [1] A strategy σ_i is a *best response* to the opponent's strategy σ_{-i} if $u_i(\sigma_i, \sigma_{-i} \geq u_i(\sigma_i', \sigma_{-i})$ for all $\sigma_i' \in \Pi_i$. Again, let $\mathcal{BR}(\sigma_{-i})$ be the set of all best responses of player i to strategy σ_{-i}.

Now, we can define the Nash and Stackelberg equilibria as with normal-form games.

Definition 2.14. [12] A strategy profile σ is a *Nash equilibrium* if for all agents i for all players i it holds that $\sigma_i \in \mathcal{BR}(\sigma_{-i})$.

Definition 2.15. [1] A strategy profile σ in an extensive-form game G is a *Strong Stackelberg equilibrium* if

$$\sigma = \underset{\sigma_1' \in \Pi_1, \sigma_2' \in \mathcal{BR}(\sigma_1')}{\operatorname{argmax}} u_1(\sigma_1', \sigma_2').$$

When commiting to mixed strategies, it may happen that the follower's best response is not uniquely determined. In such case, we assume that he breaks such ties in the leader's favor.

As an example of a Stackelberg equilibrium, consider the game in figure 2.4. In a Nash equilibrium, the leader, who plays in v_2, maximizes his outcome by playing right, receives utility 2 while giving the follower utility 0. Therefore, the follower plays right, and the leader's node is never reached, giving the leader utility 0. However, if we consider a Stackelberg equilibrium, the leader can commit to playing left and right with probability $\frac{1}{2}$. This secures both players utility 1. The follower, breaking ties in the leader's favor, plays left. The expected utilities for both players are now 1.

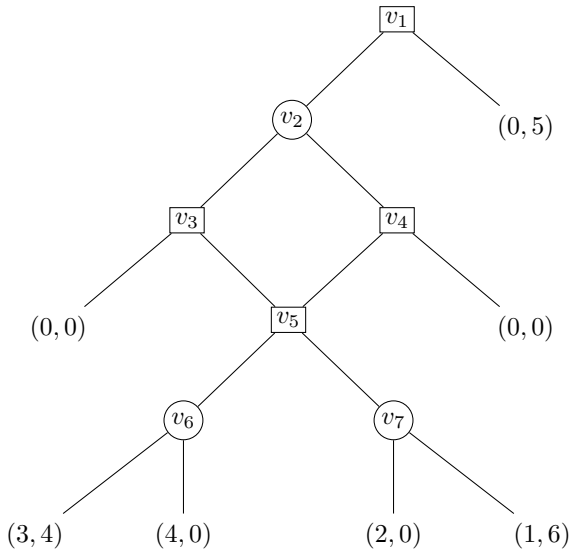

Figure 2.6. An example of a game on a DAG. The leader plays in circular nodes, the follower plays in square nodes.

In Stackelberg strategy profiles, it sometimes happens that the leader wants to commit so that the follower does not reach a certain subtree. To do so, he can choose to play a strategy which guarantees the follower his maxmin value, as it is the minimum value the leader can force the follower to obtain. The maxmin value of the follower, denoted μ, can be computed via an algorithm called bacwards induction. It is a dynamic programming pass from bottom to top. For leaves, the follower's maxmin value is simply $\mu(z) = u_2(z)$. For leader's node s, $\mu(s)$ is the minimum of $\mu(s')$, where s' is a child of s. For follower's node s, $\mu(s)$ is maximum over the maxmin values of s's children. Finally, for a chance node s the maxmin value is the weighted average of all children's maxmin values.

Before moving on to more complex classes of games, let us know present a useful visualisation of outcomes of two player games due to [8]. We can identify a leaf node with utilities (u_1, u_2) with a point in a two dimensional space. A set of all mixtures over two leaf nodes z_1, z_2 with utilities (u_1^1, u_2^1) and (u_1^2, u_2^2) is then a line segment connecting these two points. To illustrate this concept, look to Figure 2.5. We can see that the leader, who plays second, has the option of mixing between two leafs with utilities $(2, 0)$ and $(0, 2)$, hence the line segment. The vertical line positioned at $u_2 = 1$ signifies the minimum utility the leader is willing to accept when playing Left. This visualisation will become handy when computing Stackelberg equilibria of games without chance nodes.

2.2.2 Extensive-form games on DAGs

We now turn our attention to extensive form games on directed acyclic graphs. For an example game, look in Figure 2.6. As we can see, we no longer have the guarantee of having a unique history for every node in the game. This poses problems with definition of behavioral strategies. One option is to define a single strategy for every node. However, the computation of a Stackelberg equilibrium then becomes NP-hard [7]. A second option is to define a strategy for every history π that can arrise in the game. This effectively transforms the DAG into a tree, with states being histories. The equivalent tree form of our example game is in Figure 2.7. However, such trees

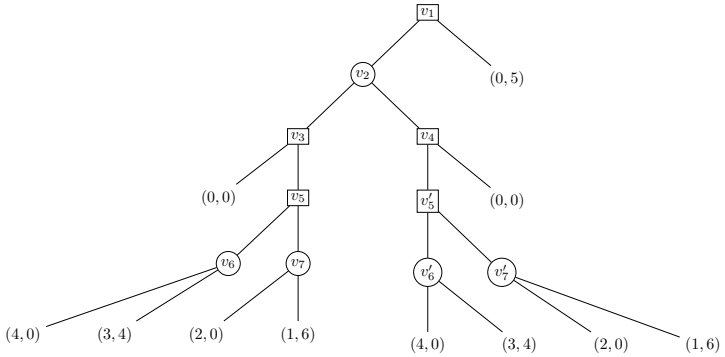

Figure 2.7. A tree equivalent of the game in Figure 2.6.

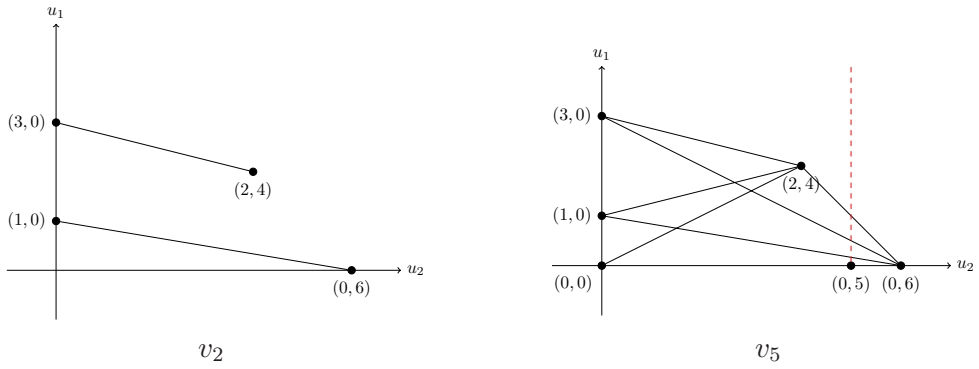

Figure 2.8. A visualisation of achievable utilities in nodes v_2 and v_5 of the game in Figure 2.6. Vertical red line signifies the maxmin value of the follower in node v_1.

can become exponentially large. There is, however, a middle ground – strategies with memory. Before we get to them, let us define strategy profiles on paths. Let P be the set of all valid paths in the game. Then

Definition 2.16. A *behavioral strategy* σ_i of player i is a collection $(\sigma_i(\pi,\cdot))_{\pi\in P}$ of independent probability measures, where $\sigma_i(\pi,\cdot)$ is a probability measure over $\mathcal{A}(s)$, s is the last node of π and $\sigma_i(s,a)$ is the probability of playing action a in s.

Again, because the player in π is uniquely determined by the last node of π, we drop the index and write $\sigma(\pi,a)$. The expected utility of such a strategy can be computed analogously to the extensive-form case. For every path $\pi = v_0a_0\dots v_k$, we can define it's probability in a strategy profile as $\sigma(\pi) = \Pi_{i=0}^{k-1}\sigma(\pi[i],a_i)$. For each leaf z, let the probability of reaching z when following σ be $p^\sigma(z)$. We can then define the expected utility of σ as $u_i(\sigma) = \sum_{z\in\mathcal{Z}} p^\sigma(z)u_i(z)$. Also, let $u^\sigma(\pi)$ be the expected utility obtained by fixing π and then continuing to play according to σ. That is, $u^\sigma(\pi)$ represents the utility the players can expect to obtain after π has already happened. Lastly, let $\sigma(\pi\downarrow)$ be the strategy in the whole subgraph defined by π.

With this definition in place, the definiton of Stackelberg equilibrium is completely identical to the case of games on trees. To see an example of such a Stackelberg equilibrium, consider the game in Figure 2.6. To visualize the utilities more clearly, in Figure 2.8 are the visualisations of achievable utilities of nodes v_2 and v_5. In the equilibrium strategy, the leader wants to make the follower play left in v_1, as he has a better utility anywhere else. In v_2, he gains the option to randomize between the leaves with utilities $(2,4)$ and $(0,6)$, an option he did not have before. Therefore, when playing left, he wants to commit so as to reach the leaf $(2,4)$. To do so, he commits to punish

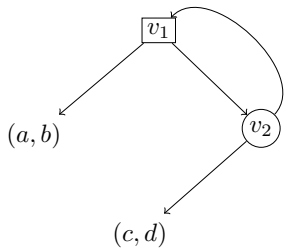

Figure 2.9. An example of a stochastic game. The leader plays in circular nodes, the follower plays in square nodes.

the leader if he chooses to play to v_6, playing to leaf $(1, 0)$. Similarly, when playing right in v_1, he commits to playing $(3, 0)$ in v_7. Thus, the follower's strategy in v_5 depends on the leader's decision in v_2. The expected utility of this equilibrium is $(1, 5)$. Note that it is the intersection between a line representing mixture and a line representing maxmin value. This property will later be used when computing Stackelberg equilibria.

We now turn our focus to strategies with memory. When playing according to a strategy with memory, players do not make decisions according to whole histories, but according to memory state. Every time an action is played in some node, the memory state is allowed to change. Thus, this memory model is an outputless finite state machine.

Definition 2.17. [3] A *strategy with memory* is a tuple $(\sigma, M, \mathcal{M}, m_0)$, where

- $\sigma = (\sigma_1, \sigma_2)$ are the strategies of players, such that $\sigma_i : M \times (\mathcal{S} \times \mathcal{A}) \to [0, 1]$ is a probability distribution on every state $s \in S_i$, such that $\sigma_i(m; s, a)$ is the probability of playing action a in state s while the memory state is m,
- M is a set of memory states
- $\mathcal{M} : M \times (\mathcal{S} \times \mathcal{A}) \to M$ is the memory update function, where $\mathcal{M}(m; s, a)$ is the memory state to which the memory changes if action a is played in state s when the memory state is m and
- $m_0 \in M$ is the initial memory state.

The expected utility of such a strategy can be once again expressed as $u_i(\sigma) = \sum_{z \in \mathcal{Z}} p^\sigma(z) u_i(z)$, where $p^\sigma(z)$ is the probability of reaching z if playing according to sigma while changing the memory according to $\mathcal{M}$. Consider the example game in figure 2.6. The Stackelberg equilibrium we described before can be easilly represented by a strategy with two memory states $M = \{m_0, m_1\}$. We can start in m_0, and in v_2 set $\mathcal{M}(m_0; v_2, Left) = m_0$ and $\mathcal{M}(m_0; v_2, Right) = m_1$. All other transitions of $\mathcal{M}$ just keep the same state. Doing so allows us to represent the strategy in v_5 by two memory states. In chapter Results, we find strategies with memory for which the memory set M is bounded by the size of the game tree, thus providing exponentially smaller representation of strategies.

2.2.3 Stochastic games

If the sequential game is played on a general graph with cycles, it is called a *stochastic game*. In this thesis, we focus on stochastic games without chance nodes only. To see an example of a stochastic game, refer to Figure 2.9. As before, the stochastic games can have a tree form, although an infinite one, as in Figure 2.10. To avoid problems with infinite cycles, we consider *indefinite horizon games*. In an indefinite horizon game, there has to exist a number k (although not known beforehand) such that no history

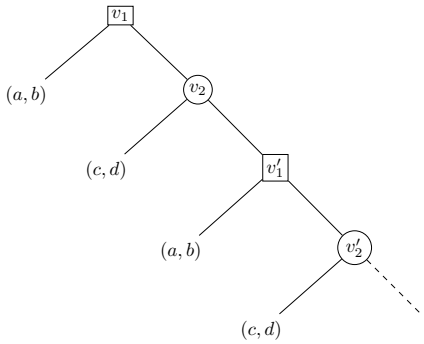

Figure 2.10. A tree equivalent of the game in Figure 2.9. To the right of v_2', the tree continues to infinitely repeat itself.

π of length at least k is reached with probability greater than zero. This allows us to keep formalism from games on DAGs.

2.3 Solving Stackelberg equilibria

In this section we turn our focus to known algorithms for finding Stackelberg equilibria. First, we consider games on trees without chance nodes. For such games, Letchford and Conitzer [8] provided a polynomial time algorithm. For games on trees with chance nodes, Letchford proved that finding a solution is NP-hard [7]. However, there is an approximation algorithm that can find an additive approximation for arbitrary ϵ [1].

2.3.1 Trees without chance nodes

Theorem 2.2. [8] In a two-player extensive-form game without chance nodes, the Stackelberg equilibrium can be found in $O(|\mathcal{S}_2|\,|\mathcal{S}_1|^2)$.

The algorithm consists of three dynamic programming passes. First, it will compute the follower's maxmin value for every node v using backwards induction. Second, it will traverse the tree from bottom to top, computing the set S_n of possible mixtures over leaf nodes reachable from n. It will do so by considering these mixtures as points in the utility space (see 2.8). Third, there will be a top to bottom pass determinign the optimal strategy σ.

Upward pass Because we are interested in finding the strategy that maximizes the leaders utility, it is sufficient to find the upper envelope of S_n, and not the whole set. For each node n, we will therefore construct two sets. A set S_n^1 will contain line segments in the form of (p_1, p_2). It will contain line segments which form the upper envelope of S, and some more segments for computational reasons. A second set, S_n^2 will contain the ending points of some, but not all, segments in S_n^1.

In a leaf node l, simply set $S_l^1 = \{(l,l)\}$ and $S_l^2 = \{l\}$. In a node n in which the leader plays, he has two options. He can either commit to a pure strategy, thus being able to reach any point from any of his children. Or, he can play a mixed strategy. Because we have a two player game, it is never optimal to mix between more than two actions (as such strategies would be dominated). Therefore, for every pair of children u, v of n, the leader can achieve points on all line segments which have one end in S_u^2 and the other and in S_v^2. This gives us $S_n^2 = \bigcup_{v \in children(n)} S_v^2$ and $S_n^1 = (\bigcup_{v \in children(n)} S_n^1) \cup \{(p,p') | p \in S_v^2; p' \in S_w^2; v, w \in children(n); v < w\}$. Note that in order to avoid duplicates, arbitrary ordering of nodes is assumed.

As for the follower's nodes, it may happen that not all outcomes of followers node n can be reached. For any action the follower will not accept any outcome in which his utility would be less than the best minimum utility he can get if he plays another action. Formally, let v be a child of n and let $\mu(v)$ be the maxmin value of v for the follower. Note that we can get this value for free by taking the minimal follower's utility over nodes in S_v^2, as S_v^2 always contains the leftmost point of segments in S_v^1. Let $i_n(v) = \max_{w \neq v, w \in children(n)} u_2(\mu(n))$. This is the lowest utility that the follower will accept, if he is to play to v (as he is always guaranteed to get at least $i_n(v)$ elsewhere). Therefore, we can construct the reachable line segments from S_v^1 by cutting them vertically at $i_n(v)$. We will do this in two steps. First, we will create sets $\hat{S}_v^1$ of cut line segments and sets $\hat{S}_v^2$ of newly generated line endpoints. Second, we then find which endpoints from $\hat{S}_v^2$ we actually need to keep.

For each child v of n and each line segment $(p, q) \in S_v^1$, if:

- $u_2(p) \geq i_n(v)$ and $u_2(q) \geq i_n(v)$, add (p, q) to $\hat{S}_v^1$,
- $u_2(p) < i_n(v)$ and $u_2(q) < i_n(v)$, throw the segment away,
- $u_2(p) < i_n(v)$ and $u_2(q) \geq i_n(v)$, cut (p, q) at $i_n(v)$, that is find λ such that $u_2(\lambda p + (1-\lambda)q) = i_n(v)$ and add $(\lambda p + (1-\lambda)q, q)$ to $\hat{S}_v^1$ and $\lambda p + (1-\lambda)q$ to $\hat{S}_v^2$,
- $u_2(p) \geq i_n(v)$ and $u_2(q) < i_n(v)$, swap p and q and repeat the point above.

Now, let $S_n^1 = \bigcup_{v \in children(n)} \hat{S}_v^1$. As for the line endpoints, we know that the cuts were done at $i_n(v)$, which is defined as maximum of maxmin value over all other children of n except v. This means, that for all but one (the maximizer) this value is the same. Let us call this maximizing child m_n. From all children of n except for m_n, we need to keep only one endpoint a - the one with the highest utility of the leader (as mixtures with this endpoint will dominate mixtures from the endpoints bellow it), that is $a \in \operatorname{argmax}_{i \in \bigcup_{w \neq m_n} \hat{S}_w^2} u_2(i)$. The second point b will be the one in $\hat{S}_{m_n}^2$ maximizing the leader's utility, $b \in \operatorname{argmax}_{i \in \hat{S}_{m_n}^2} u_2(i)$. Therefore, set $S_n^2 = \{a, b\} \cup \bigcup_{v \in children(n)} \{p \in S_w^2 | u_2(p) \geq i_v(w)\}$.

In the root node r, we can simply obtain the value of the game as $p* \in \operatorname{argmax} i \in S_r^2 u_1(i)$.

Downward pass During the downward pass, a procedure $strategy(v, p'')$ is called to determine how to commit so that p'' results in the subtree rooted by v. If $v \in \mathcal{S}_1$, we calculate $\alpha \in [0, 1]$ and find (p, p') such that $p'' = \alpha p + (1-\alpha)p'$, find children w, w' of v such that $p \in S_w^2$ and $p' \in S_{w'}^2$ and commit to playing to w with probability α and to w' with probability $1 - \alpha$, and call $strategy(w, p)$ and $strategy(w', p')$. (Note that if $w = w'$, we can play a pure stragy to w and make just one recursive call) In a follower's node, we find a descendant w such that there is a line $(p, p') \in S_w^1$ for which $p'' = \alpha p + (1-\alpha)p'$ for some $\alpha \in [0, 1]$. Then, we call $strategy(w, p'')$ and for all other children w' of v call $strategy(w', \mu(w'))$, where $\mu(w')$ is the leader's maxmin value of w'.

2.3.2 Trees with chance nodes

A second algorithm we will use in our thesis deals with approximating optimal strategies in games with chance nodes, and is due to Bošanský et al [1]. This algorithm computes for any ϵ a strategy profile that differs from the optimal one by at most ϵ. However, it's runtime depends polynomially on the inverse of ϵ.

Theorem 2.3. [1] There is an algorithm that takes as an input an extensive form game on tree with chance nodes and leader's utilities in the range $[0, 1]$ and a parameter ϵ and computes a behavioral strategy for the leader. This strategy achieves an expected

utility that differs by at most ϵ from the utility of a Stackelberg equilibrium in behavioral strategies. The algorithm runs in time $O(\mathcal{S}(H_T/\epsilon)^3)$, where H_T is the height of the tree.

The idea of the algorithm is to compute a table for every node, such that the indicies of the table represent utilities of the leader and the entries in the table represent utilities of the follower. To achieve the desired approximation factor, the utilities in the table are scaled by $(H_T+1)/\epsilon$. We can call U the highest scaled leader's utility. We can now compute the table A_T for every subtree T. The authors of the algorithm guarantee following properties for the table:

- If $A_T[k] > -\infty$, the leader has a behavioral strategy for the game tree T that offers the follower utility $A_T[k]$ while offering at least k to the leader,
- no behavioral strategy of the leader can offer the follower strictly more than $A_T[k]$ while securing at least $k + H_T$ for the leader and
- the entries of A_T are non-increasing, and $A_T[U+1] = -\infty$.

The entries of the table are computed by following procedure.
If T is a leaf node, we can fill the table from definition as

$$A_v[k] = \begin{cases} u_2 & \text{if } k \leq u_2, \\ -\infty & \text{otherwise.} \end{cases}$$

If v is a leader's node with children L and R played with probabilities p and $1-p$, if the leader obtains a guarantee i in the child L and j in child R he obtains a guarantee $pi + (1-p)j$, while the follower gets a utility of $pA_L[i] + (1-p)A_L[j]$. Therefore,

$$A_v[k] = \max_{i,j,p} \{pA_L[i] + (1-p)A_R[j] | pi + (1-p)j \geq k\}.$$

This maximization can be done by looping over all i and j. For fixed i and j, the maximization over p is a maximum over a linear function, and thus is attained in one of the extremal feasible values.

If v is a chance node, the process is similar to the leader's node, except the mixture probability is fixed. This gives us

$$A_v[k] = \max_{i,j} \{pA_L[i] + (1-p)A_R[j] | pi + (1-p)j \geq k\}.$$

And finally, if v is a follower's node, for $A_T[k]$ he can either achieve $A_L[k]$, if the leader commits to playing follower's maxmin strategy in the right subtree, or $A_R[k]$, if the leader commits to maxmin strategy in the left subtree. This gives us

$$A_v[k] = \max\left\{A_L[k] \downarrow_{\mu(R)}, A_R[k] \downarrow_{\mu(L)}\right\},$$

where

$$x \downarrow_\mu = \begin{cases} x & \text{if } x \geq \mu, \\ -\infty & \text{otherwise.} \end{cases}$$

At each node, the table can be filled in at most $O(U^3)$, as at the worst case there are three nested loops in the case of the follower's node and chance nodes. Because the utilities were scaled by $(H_T+1)/\epsilon$, this gives us the desired runtime.

Chapter 3
Results

In this chapter we show our results. For games without chance nodes on a directed acyclic graph, we show that for every Stackelberg equilibrium there is a strategy with memory which uses no more than $|\mathcal{S}_2| + |\mathcal{Z}| + 1$ states. We present an algorithm that finds this strategy in $O(|\mathcal{S}_2|^2 |\mathcal{S}|^3)$. Then, we examine stochastic games without chance nodes on general graphs. We show that the memory bound $|\mathcal{S}_2|+|\mathcal{Z}|+1$ still holds, and show that in fact every Stackelberg equilibrium is finite. Finally, we examine games on directed acyclic graphs with chance nodes. We show that the memory bound becomes quadratic, namely $|\mathcal{S}_1|\,|\mathcal{S}_2| + 1$. We show that finding optimal strategy with memory is NP-hard. Finally, we present an approximation algorithm which find a strategy for which the expected utility is withing ϵ of the optimal one.

3.1 No chance, DAG

The first studied class of games are two player games represented by a directed acyclic graph without chance nodes. Letchford and Conitzer [8] provided a polynomial algorithm for computing Stackelberg equilibrium, if the game is represented by a tree. However, they also proved that when the game is represented by a DAG and we allow only memoryless behavioral strategies, the problem of finding a Stackelberg equilibrium is NP-hard [7]. We show that if this restriction is lifted and the players are allowed to keep some information about the past progression of the game, the problem becomes polynomial.

In Lemma 3.1, we first show that the strategy which results in the follower's maxmin value is positionally determined. Then, we examine the structure of an equilibrium strategy. We use extensively the geometric representation of outcomes of the games in the space of $2D$ utilities. We first prove some technical inequalities, and later use them to get an insight into the structure of Stackelberg equilibria. We show that for every randomized strategy there is a follower's node which justifies playing this strategy, and that for every follower's node only one node in which the leader randomizes exists. This allows us to use the follower's states as memory state labels. Finally, we present an algorithm for computing the equilibrium strategy with memory.

First, we examine the strategy realizing the follower's maxmin value, as it is an important part of Stackelberg equilibria - it represents the worst follower's utility that the leader can enforce.

Lemma 3.1. In a two player sequential game on a DAG without chance nodes , there is a pure and positionally determined strategy σ such that for every node n $u_2^\sigma(n) = \mu(n)$.

Proof: Suppose for contradiction that the leader mixes in some node n after reaching it with history $\pi = v_0 a_0 \dots v_l a_l n$, playing actions $a_1^n, \dots, a_k^n$ with probabilities $p_1, \dots, p_k$. The expected utility of such strategy is $u_2^\sigma(\pi) = \sum_{i=1}^k p_i u_2^\sigma(\pi a_i^n)$. Let $m = \operatorname{argmin}_{1\le i\le k} u_2^\sigma(\pi a_i^n)$. Playing a_m^n with probability 1 has the utility $u_2^\sigma(\pi a_m^n)$. But since a_m^n is the minimizer of follower's utility, for all i we have $u_2^\sigma(\pi a_m^n) \le u_2^\sigma(\pi a_i^n)$. Hence, for the expected utility of π we obtain $u_2^\sigma(\pi) = \sum_{i=1}^k p_i u_2^\sigma(\pi a_i^n) \ge \sum_{i=1}^k p_i u_2^\sigma(\pi a_m^n) =$

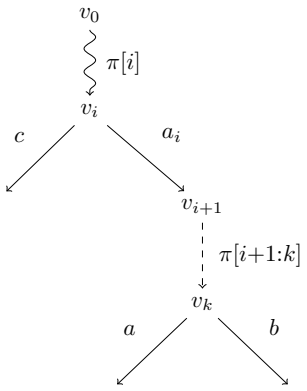

Figure 3.1. The situation where the leader mixes twice in a row, as in Lemma 3.3. The dashed line represents a pure strategy, the wavy line path $\pi[i]$.

$u_2^\sigma(\pi a_m^n)$. Therefore, by changing the strategy to playing a_m with probability 1 the follower's utility can only decrease. For the follower, who maximizes his utility, the proof is similar, except that the action with maximum utility is taken.

As for the positional determinancy, suppose that some leader's node can be reached via two histories π_1 and π_2. Without loss of generality, let $u_2^\sigma(\pi_1) \geq u_2^\sigma(\pi_2)$. Then by changing σ such that both players follow $\sigma(\pi_2)$ after π_1 the follower's utility can only decrease, thus following $\sigma(\pi_1)$ after π_1 does not achieve the follower's maxmin value. For the follower, the positional determinancy is now trivial - the leader plays positionally, and the follower always maximizes his utility. □

Now, let us examine the bound on the number of memory states which are needed to obtain the optimal strategy, and how this strategy can be found. To do so, we first prove some useful properties of Stackelberg equilibria.

Lemma 3.2. In a Stackelberg equilibrium σ in a two player sequantial game on a DAG without chance nodes , let π be a history such that the leader mixes between actions a_1 and a_2 with probabilities p and $1-p$ after π. Then either $u_1^\sigma(\pi a_1) \geq u_1^\sigma(\pi a_2)$ and $u_2^\sigma(\pi a_1) < u_2^\sigma(\pi a_2)$, or $u_1^\sigma(\pi a_1) \leq u_1^\sigma(\pi a_2)$ and $u_2^\sigma(\pi a_1) > u_2^\sigma(\pi a_2)$.

Proof: Suppose for contradiction that the leader plays such that $u_1^\sigma(\pi a_1) < u_1^\sigma(\pi a_2)$ and $u_2^\sigma(\pi a_1) < u_2^\sigma(\pi a_2)$. Then by playing a_2 with probability 1 and then following σ we obtain a strategy σ' that is strictly better for the leader and not worse for the follower, is thus dominated and not optimal. The argument reamins exactly the same for the case when $u_1^\sigma(\pi a_1) \geq u_1^\sigma(\pi a_2)$ and $u_2^\sigma(\pi a_1) \geq u_2^\sigma(\pi a_2)$ and the other direction of inequalities. □

This lemma has a nice implication in the 2D space of utilities. It tells us that the slope of the line going through the points $u^\sigma(\pi a_1)$ and $u^\sigma(\pi a_2)$ is negative, that is $slope(u^\sigma(\pi a_1), u^\sigma(\pi a_2)) \leq 0$. Note that the slope is always defined, as the inequalities for u_2 from Lemma 3.2 are strict. We can prove another useful property of slopes – when playing successive randomized strategies, the slopes of the utilities

Lemma 3.3. Let σ be a Stackelberg equilbrium in a sequential game on a DAG without chance nodes. Let there be a path $\pi = v_0 a_0 \dots v_i a_i v_{i+1} \dots v_k$ such that the leader randomizes in v_i between actions a_i and c, the strategy between v_{i+1} and v_k is pure ($\sigma(\pi[i+1{:}k]) = 1$) and in v_k he randomizes between actions a and b. Let $u_1^\sigma(\pi[k]a) \leq u_1^\sigma(\pi[k]b)$ and $u_2^\sigma(\pi[k]a) > u_2^\sigma(\pi[k]b)$. Then

$$slope(u^\sigma(\pi[k]a), u^\sigma(\pi[k]b)) \leq slope(u^\sigma(\pi[k]), u^\sigma(\pi[i]c)).$$

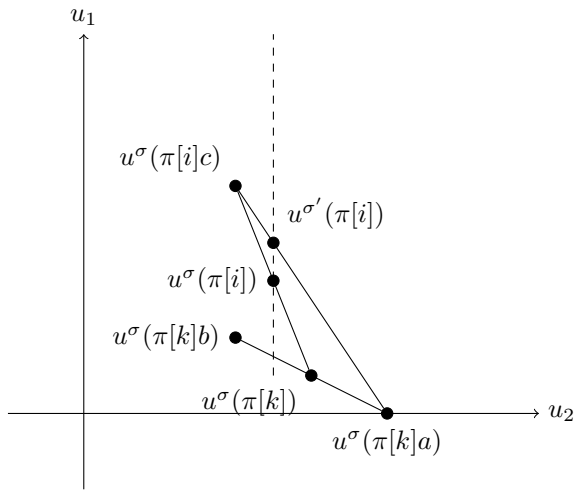

Figure 3.2. The situation from Lemma 3.3 for the case $u_2^\sigma(\pi[i]c) \leq u_2^\sigma(\pi[k])$. The dashed line represent the follower's utility of both the old strategy σ and the new strategy σ' at $\pi[i]$.

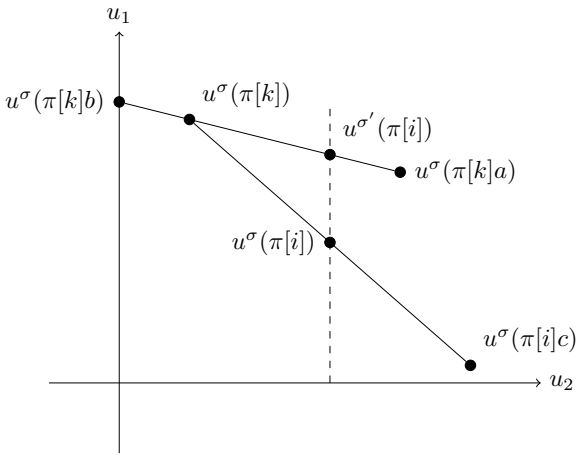

Figure 3.3. The situation from Lemma 3.3 for the case $u_2^\sigma(\pi[k]) < u_2^\sigma(\pi[i]) \leq u_2^\sigma(\pi[k]a)$. The dashed line represent the follower's utility of both the old strategy σ and the new strategy σ' at $\pi[i]$.

Proof: The situation in which the leader mixes in two states is depicted in Figure 3.1. Suppose for contradiction that

$$slope(u^\sigma(\pi[k]a), u^\sigma(\pi[k]b)) > slope(u^\sigma(\pi[k]), u^\sigma(\pi[i]c)).$$

We split the proof to three cases. Either $u_2^\sigma(\pi[i]c) \leq u_2^\sigma(\pi[k])$, $u_2^\sigma(\pi[k]) < u_2^\sigma(\pi[i]c) \leq u_2^\sigma(\pi[k]b)$ or $u_2^\sigma(\pi[k]b) < u_2^\sigma(\pi[i]c)$. In each of these cases, we will show that if the inequality does not hold, there is a strategy σ' whitch is better for the leader. Also, note that because slopes in an equilibrium are always negative, for the absolute values it holds that

$$|slope(u^\sigma(\pi[k]a), u^\sigma(\pi[k]b))| \geq |slope(u^\sigma(\pi[k]), u^\sigma(\pi[i]c))|\,.$$

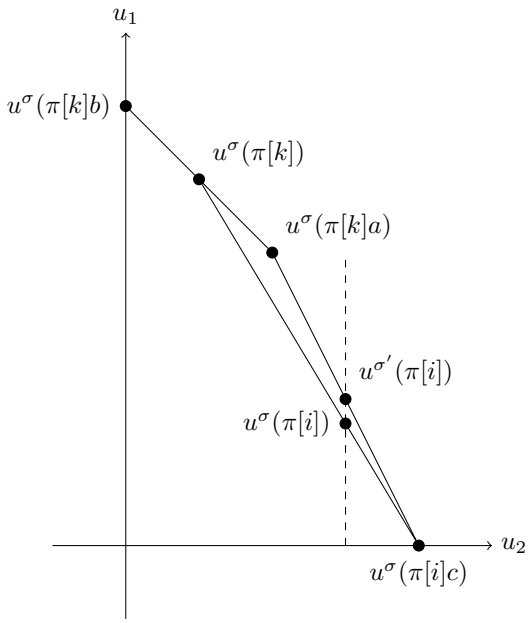

Figure 3.4. The situation from Lemma 3.3 for the case $u_2^\sigma(\pi[k]) < u_2^\sigma(\pi[i]) \leq u_2^\sigma(\pi[k]a)$. The dashed line represent the follower's utility of both the old strategy σ and the new strategy σ' at $\pi[i]$.

Case 1: $u_2^\sigma(\pi[i]c) \leq u_2^\sigma(\pi[k])$. Our aim to find a better strategy is depicted in Figure 3.2. We can find γ such that

$$\gamma u_2^\sigma(\pi[i]c) + (1-\gamma)u_2^\sigma(\pi[k]a) = u_2^\sigma(\pi[i]),$$

and set $\sigma'(\pi[i], c) = \gamma$, $\sigma'(\pi[i]a_i) = 1-\gamma$ and $\sigma'(\pi[k], a) = 1$, copying σ everywhere else. Because $\sigma(\pi[i+1{:}k]) = 1$,

$$u^{\sigma'}(\pi[i]) = \gamma u^\sigma(\pi[i]c) + (1-\gamma)u^\sigma(\pi[i]a).$$

Because $u_2^\sigma(\pi[k]a) > u_2^\sigma(\pi[k]b)$, we have $u_2^{\sigma'}(\pi[k]) > u_2^\sigma(\pi[k])$ and the follower is offered strictly more in all v_l, $i < l < k$ and does not deviate from σ'. In v_i, we have defined the strategy so that $u_2^{\sigma'}(\pi[i]) = u_2^\sigma(\pi[i])$ and the follower does not deviate in any $\pi[l]$, $l < i$. As for the leader, because

$$slope(u^\sigma(\pi[k]a, u^\sigma(\pi[k]b)) > slope(u^\sigma(\pi[k]), u^\sigma(\pi[i]c))$$

and $u^{\sigma'}(\pi[i])$ lies on the line connecting $u^\sigma(\pi[i]c)$ and $u^\sigma(\pi[k]a)$, we have $u_1^{\sigma'}(\pi[i]) > u_1^\sigma(\pi[i])$ and σ' is strictly preffered by the leader.

Case 2: $u_2^\sigma(\pi[k]) < u_2^\sigma(\pi[i]) \leq u_2^\sigma(\pi[k]a)$, as in Figure 3.3. We can find γ such that

$$\gamma u_2^\sigma(\pi[k]a) + (1-\gamma)u_2^\sigma(\pi[k]b) = u_2^\sigma(\pi[i]),$$

and set $\sigma'(\pi[k], a) = \gamma$, $\sigma'(\pi[k], b) = 1-\gamma$ and $\sigma'(\pi[i], a_i) = 1$ and copy σ everywhere else. Because $\sigma(\pi[i+1{:}k]) = 1$, we have

$$u^{\sigma'}(\pi[i]) = \gamma u^\sigma(\pi[i]a) + (1-\gamma)u^\sigma(\pi[i]b).$$

Because $u_2^\sigma(\pi[i]) > u_2^\sigma(\pi[k])$, we know that $u_2^{\sigma'}(\pi[k]) = u_2^\sigma(\pi[i]) > u_2^\sigma(\pi[k])$. The follower is therefore offered strictly more in all v_l, $i < l < k$ and does not deviate from

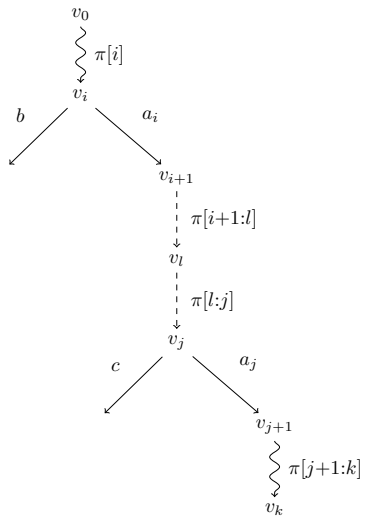

Figure 3.5. The situation where the leader mixes twice in a row, as in Lemma 3.5. The dashed lines represent pure strategies, the wavy line path with some nonzero probability. The node v_l is the follower's node found in Lemma 3.5.

σ'. Again, by definition of σ' we have $u_2^{\sigma'}(\pi[i]) = u_2^{\sigma}(\pi[i])$ and the follower does not deviate in any $\pi[l]$, $l < i$. Because

$$slope(u^{\sigma}(\pi[k]a), u^{\sigma}(\pi[k]b)) > slope(u^{\sigma}(\pi[k]), u^{\sigma}(\pi[i]c)),$$

$u_2^{\sigma}(\pi[i]c) \geq u_2^{\sigma}(\pi[i]) > u_2^{\sigma}(\pi[k])$ and $u^{\sigma'}(\pi[i])$ lies on the line between $u^{\sigma}(\pi[k]a)$ and $u^{\sigma}(\pi[k]b)$, it holds that $u_1^{\sigma'}(\pi[i]) > u_1^{\sigma}(\pi[i])$. Thus, σ' is strictly preffered by the leader.

Case 3: $u_2\sigma(\pi[i]) > u_2^{\sigma}(\pi[k]a)$, as in Figure 3.4. We can find γ such that

$$\gamma u_2^{\sigma}(\pi[k]a) + (1-\gamma)u_2^{\sigma}(\pi[i]c),$$

and set $\sigma'(\pi[k], a) = 1$, $\sigma'(\pi[i], a_i) = \gamma$ and $\sigma'(\pi[i], c) = 1-\gamma$. Because $\sigma(\pi[i{+}1{:}k]) = 1$, we have

$$u^{\sigma'}(\pi[i]) = \gamma u^{\sigma}(\pi[i]a) + (1-\gamma)u^{\sigma}(\pi[i]c).$$

By the same argument as in the first case, the follower does not deviate from σ' and the leader strictly preffers σ'. □

It will be useful later to limit ourselves to strategies for which the inequality from Lemma 3.3 is strict.

Lemma 3.4. Let σ be a Stackelberg equilbrium in a sequential game on a DAG without chance nodes. Let there be a path $\pi = v_0a_0 \dots v_ia_iv_{i+1} \dots v_k$ such that the leader randomizes in v_i between actions a_i and c, the strategy between v_{i+1} and v_k is pure ($\sigma(\pi[i+1{:}k]) = 1$) and in v_k he randomizes between actions a and b. Let $u_1^{\sigma}(\pi[k]a) \leq u_1^{\sigma}(\pi[k]b)$ and $u_2^{\sigma}(\pi[k]a) > u_2^{\sigma}(\pi[k]b)$ and $slope(u^{\sigma}(\pi[k]a), u^{\sigma}(\pi[k]b)) = slope(u^{\sigma}(\pi[k]), u^{\sigma}(\pi[i]c))$. Then there exists an equilibrium σ' with $u^{\sigma} = u^{\sigma'}$ in which the leader randomizes in at most one of $\pi[i]$ and $\pi[k]$.

Proof: The proof is analogous to the proof of Lemma 3.3. We consider three cases, $u_2^{\sigma}(\pi[i]c) \leq u_2^{\sigma}(\pi[k])$, $u_2^{\sigma}(\pi[k]) < u_2^{\sigma}(\pi[i]c) \leq u_2^{\sigma}(\pi[k]b)$ or $u_2^{\sigma}(\pi[k]b) < u_2^{\sigma}(\pi[i]c)$. In each of such case, the argument from proof of Lemma 3.3 holds, giving us an equilibrium σ' with $u^{\sigma'}(\pi[i]) = u^{\sigma}(\pi[i])$ and thus (as $\sigma = \sigma'$ everywhere else) $u^{\sigma'} = u^{\sigma}$. □

From now on, we therefore limit ourselves to strategy profiles in which the inequality in Lemma 3.3 is strict.

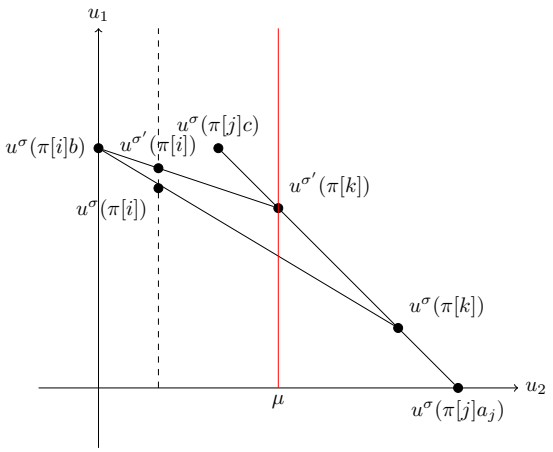

Figure 3.6. The situation from Lemma 3.5 for the case $u_2^\sigma(\pi[i]) \leq \mu \leq u_2^\sigma(\pi[j])$. The dashed line represent the expected utility of the strategy being changed.

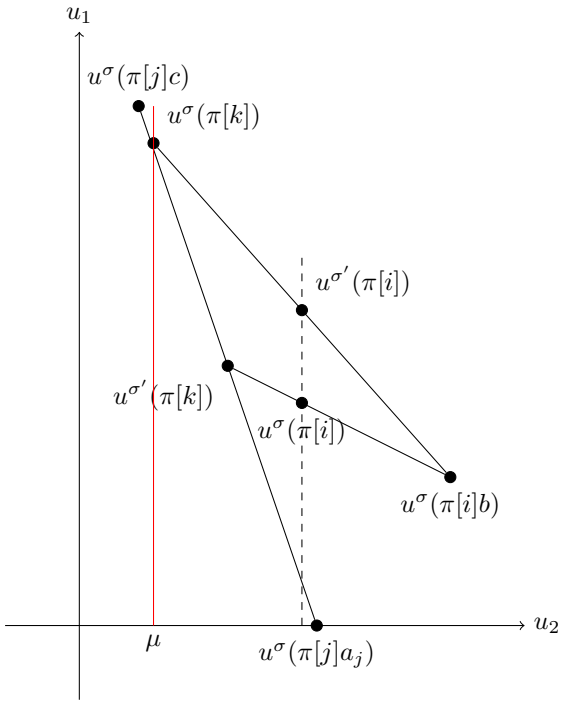

Figure 3.7. The situation from Lemma 3.5 for the case if $\mu \leq u_2^\sigma(\pi[j]) \leq u_2^\sigma(\pi[j]c)$. The dashed line represent the expected utility of the strategy being changed. The red line is the value of μ.

Now we come to reasoning about the structure of equilibria. We show that between two nodes in which the leader randomizes there is a follower's state for which the maxmin value is attained. In other words, the follower's states justifies playing the second mixture. This will leave us with all mixtures justified except for the first one on any path. With such mixtures we will deal later.

Lemma 3.5. Let σ be a Stackelberg equilibrium in a sequential game on a DAG without chance nodes. Let $\pi = v0a_0 \dots v_{k-1}a_{k-1}v_k$ be a history played with nonzero probability

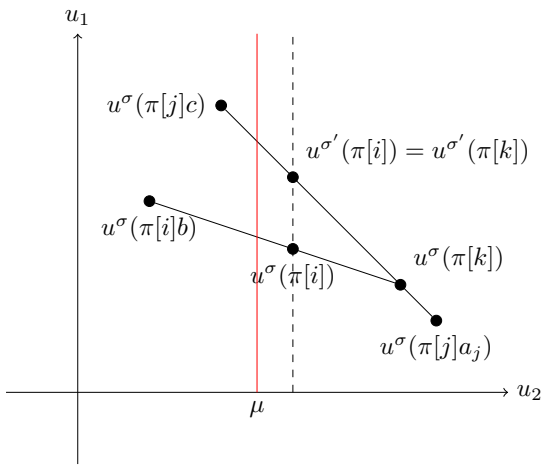

Figure 3.8. The situation from Lemma 3.5 when $\mu < u_2^\sigma(\pi[i]) < u_2^\sigma(\pi[j])$. The dashed line represent the expected utility of the strategy being changed. The red line is the value of μ.

in σ. Then for every $0 \leq i < j < k$ such that the leader randomizes after histories $\pi[i]$ (between actions a_i and b with probabilites α and $1-\alpha$) and $\pi[j]$ (between a_j and c with probabilities β and $1-\beta$) and $\sigma(\pi[i+1{:}j]) = 1$ there exists an $i < l < k$ such that the leader plays in $\pi[l]$ and $u_2^\sigma(\pi[l]) = \mu(v_l)$.

Proof: The situation is depicted in Figure 3.5. Suppose for contradiction that there is a history π and indices i, j such that the leader randomizes after $\pi[i]$ and $\pi[j]$, $\sigma(\pi[i+1{:}j]) = 1$ but for every $i < l < k$ such that the follower plays $u_2^\sigma(\pi[l]) \neq \mu(v_l)$. If for some l $u_2^\sigma(\pi[l]) < \mu(v_l)$, the follower has an incentive to deviate as he can always achieve utility at least $\mu(v_l)$ in v_l. Therefore, for every $i < l < j$ it holds that $u_2^\sigma(\pi[l]) > \mu(v_l)$. We will find a strategy that has better utility for the leader and the same utility for the follower, and for which there exists a follower's node with desired properties. We aim to find γ and δ such that the leader can play a with probability γ, b with probability $1-\gamma$, a_i with probability δ and c with probability $1-\delta$. By copying σ everywhere else we obtain a new strategy σ'. Our aim is depicted in the space of utilities in Figures 3.6, 3.8 and 3.7.

From Lemma 3.3, we know that

$$slope(u^\sigma(\pi[j+1]), u^\sigma(\pi[j]c)) < slope(u^\sigma(\pi[i]b), u^\sigma(\pi[i+1]))$$

. By Lemma 3.2, let $u_2^\sigma(\pi[j+1]) > u_2^\sigma(\pi[j]c)$ and $u_1^\sigma(\pi[j+1]) \leq u_1^\sigma(\pi[j]c)$. Also note that because $\sigma(\pi[i+1{:}j]) = 1$, $u^\sigma(\pi[i+1]) = u^\sigma(\pi[j])$. If there is no follower's node between v_i and v_j, mixing both in v_i and v_j is equivalent to mixing between three nodes and therefore not optimal. If there is at least one follower's node, we will find μ such that the follower does not deviate in any $\pi[l], i < l < j$ if offered utility at least μ. Let $\mu' = \max_{i<l<j} \mu(\pi[l])$ and $\mu = \max\{\mu', u_2^\sigma(\pi[j]c)\}$. Because for all l $u_2^\sigma(\pi[l]) > \mu(v_l)$, it also holds that $u_2^\sigma(\pi[l]) > \mu'$ and thus $u_2^\sigma(\pi[k]) > \mu'$. Also, because

$$u_2^\sigma(\pi[k]) = \beta u_2^\sigma(\pi[j+1]) + (1-\beta) u_2^\sigma(\pi[j]c),$$

$0 < \alpha < 1$ and $u_2^\sigma(\pi[j+1]) > u_2^\sigma(\pi[j]c)$, it holds that $u_2^\sigma(\pi[l]) > \mu$.

We will now proceed with two cases. For the first case, either $u_2^\sigma(\pi[i]b) \leq \mu \leq u_2^\sigma(\pi[j])$ or $\mu \leq u_2^\sigma(\pi[j]) \leq u_2^\sigma(\pi[j]c)$. For the second case, we deal with $\mu < u_2^\sigma(\pi[i]b) < u_2^\sigma(\pi[j])$.

Case 1: $u_2^\sigma(\pi[i]) \leq \mu \leq u_2^\sigma(\pi[j])$ or $\mu \leq u_2^\sigma(\pi[j]) \leq u_2^\sigma(\pi[j]c)$. This corresponds to Figures 3.6 and 3.7. We can find an γ such that

$$\gamma u_2^\sigma(\pi[j+1]) + (1-\gamma)u_2^\sigma(\pi[j]c) = \mu$$

and set $\sigma'(v_j, a_j) = \gamma$ and $\sigma'(v_j, c) = 1-\gamma$. Thus

$$u^{\sigma'}(\pi[j]) = \gamma u^\sigma(\pi[j+1]) + (1-\gamma)u^\sigma(\pi[j]c).$$

This allows us to find δ such that

$$u_2^\sigma(\pi[i]) = \delta u_2^{\sigma'}(\pi[j]) + (1-\delta)u_2^\sigma(\pi[i]b),$$

and set $\sigma'(v_i, a_i) = \delta$ and $\sigma'(v_i, b) = 1-\delta$. Thus

$$u^{\sigma'}(\pi[i]) = \delta u^{\sigma'}(\pi[j]) + (1-\delta)u^\sigma(\pi[i]b).$$

We copy σ everywhere else.

We defined σ' such that $u_2^{\sigma'}(\pi[i]) = u_2^\sigma(\pi[i])$, therefore the follower will not deviate in some $l < i$. However, we need to see that he is not incentivized to deviate in some v_l, $i < l < j$. From the definition of σ', we have $u_2^{\sigma'}(\pi[l]) = u_2^{\sigma'}(\pi[j]) = \mu$. But from the definition of μ we have $\mu \geq \mu(\pi[l])$, and thus $u_2^{\sigma'}(\pi[l]) \geq \mu(\pi[l])$ and the follower is not incentivized to deviate. We now just need to ensure that the leader will obtain better utility by playing σ'. I holds that

$$slope(u^\sigma(\pi[j+1]), u^\sigma(\pi[j]c) < slope(u^\sigma(\pi[i]b), u^\sigma(\pi[i+1]))$$

and both $u^\sigma(\pi[j])$ and $u^{\sigma'}(\pi[j])$ lie on the line between $u^\sigma(\pi[j+1])$ and $u^\sigma(\pi[j]c)$. But because $u_2^{\sigma'}(\pi[j]) \leq u_2^\sigma(\pi[j])$, for any point p' on the line segment between $u^{\sigma'}(\pi[j])$ and $u^\sigma(\pi[i]b)$ and any point p on the line segment between $u^\sigma(\pi[j])$ and $u^\sigma(\pi[i])b$ it holds that $p'_1 > p_1$. Hence $u_1^{\sigma'}(\pi[i]) > u_1^\sigma(\pi[i])$ and σ' is strictly preffered by the leader.

Case 2: $\mu < u_2^\sigma(\pi[i]) < u_2^\sigma(\pi[j])$, as in Figure 3.8. We can find δ such that

$$u_2^\sigma(\pi[i]b) = \delta u_2^\sigma(\pi[j+1]) + (1-\delta)u_2^\sigma(\pi[j]c)$$

and set $\sigma'(\pi[j], a_j) = \delta$, $\sigma'(\pi[j], c) = 1-\delta$ and $\sigma'(\pi[i], a_i) = 1$. Because the strategy between v_i and v_j is now pure ($\sigma'(\pi[i{:}j] = 1)$, we have

$$u^{\sigma'}(\pi[i]) = u^{\sigma'}(\pi[j]) = \delta u^\sigma(\pi[j+1]) + (1-\delta)u^\sigma(\pi[j]c).$$

From the definition of σ' we again know that the follower does not deviate in any v_l, $l < i$. For all $i < l < j$, we know that $u_2^{\sigma'}(\pi[l]) = u_2^\sigma(\pi[i]b) > \mu$. This gives us again $u_2^{\sigma'}(\pi[l]) > \mu(v_l)$ and the follower does not deviate. As ford the leader, because

$$slope(u^\sigma(\pi[j+1]), u^\sigma(\pi[j]c) < slope(u^\sigma(\pi[i]b), u^\sigma(\pi[i+1]))$$

and $u^{\sigma'}(\pi[i])$ now lies on the line segment between $u^\sigma(\pi[j]c)$ and $u^\sigma(\pi[j+1])$, necessarilly $u_1^{\sigma'}(\pi[i]) > u_1^\sigma(\pi[i])$ and σ' is again strictly preffered by the leader. □

The previous lemma lefts unadressed what happens before the leader mixes for the first time. To solve this, we prove the following lemma, which contains no new ideas from before. To visualize the situation, we encourage the reader to refer to Figure 3.5 and imagine that the node v_{i+1} is in fact the root node. From this visualisation we can also see that the proof method will remain exactly the same.

Lemma 3.6. Let σ be a Stackelberg equilibrium in a sequantial game on a DAG without chance nodes. Let $\pi = v_0 a_0 \dots v_k a_k v_{k+1}$ be a history such that $\sigma(\pi[k-1]) = 1$ and the leader randomizes in v_k between actions a_k and b with probabilities α and $1-\alpha$. Then there exists a $j < k$ such that the follower plays in v_j and $u_2^\sigma(\pi[j]) = \mu(v_j)$.

Proof: Suppose for contradiction that there is a history π in which for all $j < k$ such that the follower plays in v_j $u_2^\sigma(\pi[j]) \neq \mu(v_j)$. From Lemma 3.3, let $u_2^\sigma(\pi[k+1]) > u_2^\sigma(\pi[k]b)$ and $u_1^\sigma(\pi[k+1]) \leq u_1^\sigma(\pi[k]b)$. We will find a strategy σ' which is better for the leader without incentivizing the follower to deviate. As in Lemma 3.5, we can prove that for all j $u_2^\sigma(\pi[j]) > \mu(v_j)$ and define $\mu' = \max_{i<l<j} \mu(\pi[l])$ and $\mu = \max\{\mu', u_2^\sigma(\pi[j]c)\}$. We can find $\beta \leq \alpha$ such that

$$\mu = \beta u_2^\sigma(\pi[k+1]) + (1-\beta) u_2^\sigma(\pi[k]b)$$

and define $\sigma'(\pi[k], a_k) = \beta$ and $\sigma'(\pi[k], b) = 1-\beta$. As in Lemma 3.5, we obtain $u_2^{\sigma'}(\pi[j]) \geq \mu(\pi[j])$ for every $j < k$ and the follower does not deviate. Because $u_1^\sigma(\pi[k+1]) \leq u_1^\sigma(\pi[k]b)$ and $\beta \leq \alpha$, we have $u_1^{\sigma'}(\pi[k]) \geq u_1^\sigma(\pi[k])$ and σ' is no worse than σ. □

Up until now, we were finding a follower's node for every randomized strategy we could think of. Now, we focus our attention the other way. We now examine what happens when a follower's node n is reached. We show that when it is reached via two paths such that the leader commits to offering the leader utility $\mu(n)$, the strategy for the to paths has to be identical. This will then leave us in an excellent position to define memory states - for every follower's state, we will have at most one mixed strategy, and for every mixed strategy a follower's node.

Lemma 3.7. Let σ be a Stackelberg equilibrium in an extensive form game on a DAG. Let π_1, π_2 be two histories ending in a follower's node n such that $u_2^\sigma(\pi_1) = u_2^\sigma(\pi_2) = \mu(n)$. Then $u_1^\sigma(\pi_1) = u_1^\sigma(\pi_2)$.

Proof: Suppose that there are two such histories, $\pi_1 = u_0 a_0 \dots u_i a_i n$ and $\pi_2 = v_0 a_0 \dots v_j a_j n$ with $u_2^\sigma(\pi_1) = u_2^\sigma(\pi_2) = \mu(n)$, such that $u_1^\sigma(\pi_1) > u_1^\sigma(\pi_2)$. We can create a strategy σ' by copying σ everywhere except in the subgraph under $\sigma(\pi_2)$, setting $\sigma'(\pi_2 \downarrow) = \sigma(\pi_1 \downarrow)$. By doing so, the follower obtains the same utility after π_2, while the leader's utility increases. Thus, σ' is strictly preffered by the leader and σ is not an equilibrium. Because the same argument can be made for $u_1^\sigma(\pi_1) > u_1^\sigma(\pi_2)$, necessarilly $u_1^\sigma(\pi_1) = u_1^\sigma(\pi_2)$. □

This lemma tells us that for all histories π that reach some follower's node n for which $u_2^\sigma(\pi) = \mu(n)$, it suffices to consider a single strategy in all histories which follow after π. By combining this with Lemmas 3.5 and 3.6, we find that for every follower's node there is at most one leader's node in which the leader randomizes, and no other randomizations happen. Therefore, whenever the follower randomizes in v and then plays a pure strategy to w, where he randomizes again, there is a unique followers node n which identifies the path between v and w. This allows us to represent the strategy compactly using memory - we can simply keep track of the identifying followers node, or the target leaf node after the leader mixed for the last time. For strategies off the equilibrium path, we add one more memory state signifying that maxmin should be played.

Theorem 3.1. In a sequential game a DAG without chance nodes, for every Stackelberg equilibrium $\sigma : \Pi \times \mathcal{A} \to [0,1]$ there is a strategy with memory $(\sigma', M, \mathcal{M}, m_0)$ with $|M| \leq |\mathcal{S}_2| + |\mathcal{Z}| + 1$ such that $u(\sigma) = u(\sigma')$.

Proof: Let $M = \{l_v | v \in \mathcal{S}_2\} \cup \{l_z | z \in \mathcal{Z}\} \cup \{l^*\}$. By Lemmas 3.5, 3.7 and 3.6, for every path $\pi = v_0 a_0 \dots v_m a_m z$ from the root to a leaf played with nonzero probability in σ we have indices $i_1 \dots i_l$ such that $\sigma(\pi[i_k + 1{:}i_{k+1}]) = 1$, $\sigma(a_{i_k}) < 1$, $\sigma(a_{i_{k+1}}) < 1$

(the leader mixes in v_{i_k} and $v_{i_{k+1}}$) and there is a $i_k < j_k < i_{k+1}$ such that the follower plays in v_{j_k} and $u_2^\sigma(\pi[j_k]) = \mu(v_{j_k})$. Therefore, for all $1 \leq k < l$ and $i_k < n < i_{k+1}$ set

$$\begin{aligned}\mathcal{M}(l_{v_{j_k}}; v_n, a_n) &= l_{v_{j_k}}\\ \sigma'(l_{v_{j_k}}; v_n, a_n) &= 1\\ \mathcal{M}(l_{v_{j_k}}; v_{i_{k+1}}, a_{i_{k+1}}) &= l_{v_{j_{k+1}}}\\ \sigma'(l_{v_{i_k}}; v_{i_k}, a_{i_k}) &= \sigma(\pi[i_k], a_{i_k}).\end{aligned}$$

After the last index i_l, the path continues to a leaf z with $\sigma(\pi[i_l+1{:}m]) = 1$. Therefore, set

$$\begin{aligned}\mathcal{M}(l_{v_{j_{l-1}}}; v_{i_l}, a_{i_l}) &= l_z\\ \sigma'(l_{v_{j_{l-1}}}; v_{i_l}, a_{i_l}) &= \sigma(\pi[i_l], a_{i_l});\end{aligned}$$

and for all $n > i_l$ let

$$\begin{aligned}\mathcal{M}(l_z; v_n, a_n) &= l_z\\ \sigma'(l_z; v_n, a_n) &= 1.\end{aligned}$$

Before the first index i_1, the leader plays a pure strategy (as i_1 is the index of the first mixture). Due to Lemma 3.6, there is an index j_0 such that the follower plays in v_{j_0} and $u_2^\sigma(\pi[j_0]) = \mu(v_{j_0})$. Thus, set

$$\begin{aligned}\mathcal{M}(l_{v_{j_0}}; v_{i_1}, a_{i_1}) &= l_{v_{j_1}}\\ \sigma'(l_{v_{j_0}}; v_{i_1}, a_{i_1}) &= \sigma(\pi[i_1], a_{i_1})\end{aligned}$$

and for all $n < i_1$ let

$$\begin{aligned}\mathcal{M}(l_{v_{j_0}}; v_n, a_n) &= l_{v_{j_0}}\\ \sigma'(l_{v_{j_0}}; v_n, a_n) &= 1.\end{aligned}$$

Set all undefined transitions to $\mathcal{M}(m; v, a) = l^*$, so that deviations end up in the strategy realizing follower's maxmin value. Finally, for every node $\sigma'(l^*; n, a) = \sigma^\mu(n, a)$. The initial memory state $m_0 = l_{v_{j_0}}$.

Let us now show that σ' and $\mathcal{M}$ are defined correctly. Suppose there are two histories $\pi_1 = v_0^1 a_0^1 \dots v_k^1$ and $\pi_2 = v_0^2 a_0^2 \dots a_l^2$ such that for some $i < k$ and $j < l$ we have $v_i^1 = v_j^2 = w$, the follower plays in w and $u_2^\sigma(\pi_1[i]) = \mu(w)$ and $u_2^\sigma(\pi_2[j] = \mu(w)$. By Lemma 3.7 the strategies after $\pi_1[i]$ and $\pi_2[j]$ are the same, and thus σ' and $\mathcal{M}$ are defined correctly. From Lemmas 3.5 and 3.6, we know that randomized strategies happen only after such w. Thus, the strategy σ' yields the same result as the strategy σ, with $|M| = |\mathcal{S}_2| + |\mathcal{Z}| + 1$. □

Knowing how much memory is needed and what the memory states should represent allows us to adapt algorithm of Letchford and Conitzer [8] to work on directed acyclic graphs.

Theorem 3.2. Stackelberg equilibrium with memory in a sequential on a DAG without chance nodes can be found in $O(|\mathcal{S}_2|^2\,|\mathcal{S}|^3)$.

Proof: The algorithm consists of three dynamic programming passes. First, it will compute the follower's maxmin values. Second, it will perform a dynamic programming pass identical to that in Theorem 2.2. Third, there will be a top to bottom pass determinign the optimal strategy σ, the memory update function $\mathcal{M}$ and the necessary memory labels M.

Upward pass The upward pass is identical to that in Theorem 2.2, computing the sets S_n^1 and S_n^2 for every node n, obtaining the solution $p* \in \operatorname{argmax} i \in S_r^2 u_1(i)$ in the root.

Downward pass During the downward pass, we will construct the set of memory states M, the memory update function $\mathcal{M}$ and the equilibrium strategy profile σ. For each node n we will create a set P_n. If $p \in P_n$, it means that the players should commit in p with memory state l_p (simply a label associated with p) so that p will result. We will do so layer-by-layer in the DAG, which ensures that when a node n is considered, all it's parents were already processed, and therefore the set P_n is complete. For playing strategies off the equilibrium path, we designate a special memory state l_0.

To start, set $P_r = \{p*\}$ and $M = \{p*, l_0\}$ and begin processing the root. In a node n in which the leader plays, for each $p \in P_n$, we find a line $(p', p'') \in S^1_n$ such that p lies on this line, that is find λ such that $p = \lambda p' + (1-\lambda)p''$. For both p' and p'' we find the children of n from which these points originated, that is find u and v such that $p' \in S^2_u$ and $p'' \in S^2_v$. If $u = v$, we can simply commit to playing to u and keeping the memory state the same, as we still target the same point, that is $\mathcal{M}(l_p; n, a_u) = l_p$ (where a_u is the action leading to u), $\sigma(l_p, n, a_u) = 1$. Finally, add p to P_u. If $u \neq v$, the line segment originated in n and we have to commit to mixing between u with probability λ and v with probability $1-\lambda$, while updating the memory states accordingly. That is $\sigma(l_p, n, a_u) = \lambda$, $\sigma(l_p, n, a_v) = 1-\lambda$, $\mathcal{M}(l_p; n, a_u) = l_{p'}$ and $\mathcal{M}(l_p; n, a_v) = l_{p''}$. If M does not contain $l_{p'}$ and $l_{p''}$, add them there. Finally, add p' to P_u and p'' to P_v.

In a node n in which the follower plays, we simply need to find the child v from which p came, that is find v such that there exists $(p', p'') \in S^1_v$ so that $p = \lambda p' + (1-\lambda)p''$. We then set $\sigma(l_p, n, a_v) = 1$, $\mathcal{M}(l_p; n, a_v) = l_p$ and add p to P_v. For every other child w of n, the leader wants to discourage the follower from playing there by playing maxmin. We can therefore set $\mathcal{M}(l_p; n, a_w) = l_0$.

Finally, to provide the players a maxmin strategy to default to, we set all previously undefined transitions of $\mathcal{M}$ to l_0 and compute maxmin strategy $\sigma(l_0, \cdot)$ by backwards induction (which can be done in $O(|\mathcal{S}|\,(|\mathcal{S}| + |\mathcal{Z}|)$ time).

Correctness The upward pass remains unchanged from [8]. Because every subtree under a node n in the tree version of a DAG is identical, the upward pass reamins correct. During the downward pass, the construction of $\mathcal{M}$ and M ensures that for every targetted outcome p there is a corresponding memory state, and therefore no information about the played strategy is lost as compared to the tree case. Moreover, this construction implements the scheme from Theorem 3.1.

Complexity To reason about the complexity of the algorithm, let us now first bound the sizes of S^1_n and S^2_n for each n.

In S^2_n, the endpoints may come from two different sources. They can be generated by a leaf, or by cutting some line segment of some follower's node. However, as we argued before, each follower's node may add only one new endpoint. We may thus bound the number of these endpoints both in the union of all S^2's and in each of them sepparately by $|\mathcal{S}_2| + |\mathcal{Z}|$.

As for S^1_n, the line segments may come from three sources. One, the can be generated by a leaf. Two, they can be generated in a leader's node by mixing over some two children. Because the number of points in union of all S^2's is bounded by $|\mathcal{S}_2| + |\mathcal{Z}|$, we can bound the number of lines generated by picking pairs of them by $(|\mathcal{S}_2| + |\mathcal{Z}|)^2$. And finally three, the lines can be generated by cutting another lines in a follower's node. Note that we can consider cutting only the lines that originated in a leader's node, as cutting previously cut line is equivalent to cutting the original line directly. Because at each follower's node the cut happens only at one level, we can bound the number of these lines (again in the union of all S^1's) by $|\mathcal{S}_2|\,(|\mathcal{S}_2| + |\mathcal{Z}|)^2$. The total size of S^1_n therefore lies in $O(|\mathcal{S}_2|\,(|\mathcal{S}_2| + |\mathcal{Z}|)^2)$.

During the upward pass in the leader's node n, we need to accumulate the line points from n's children. To do so, we must traverse S_v^1 for every v and for each line spend a constant time adding in to S_n^1. Because n has at most $|S|$ children, this step takes $|\mathcal{S}|\left|S^1\right|$ operations. We also have to consider all pairs of n's children and mix between the possible end points. For one pair of children this takes $(|\mathcal{S}_2|+|\mathcal{Z}|)^2$, for every pair of children this takes $|\mathcal{S}|^2(|\mathcal{S}_2|+|\mathcal{Z}|)^2$. Because we are processing every leader's node once, the time required to process them all is in $O(|\mathcal{S}_1||\mathcal{S}|^2(|\mathcal{S}_2|+|\mathcal{Z}|)^2)$.

In the followers node, we need to traverse line segments of all it's children and for each such segment decide, whether to cut it or not. Therefore, processing one child node takes $O(|\mathcal{S}||\mathcal{S}_2|(|\mathcal{S}_2|+|\mathcal{Z}|)^2)$ time, and processing all followers nodes therefore takes $O(|\mathcal{S}||\mathcal{S}_2|^2(|\mathcal{S}_2|+|\mathcal{Z}|)^2)$ The total time required for the upward pass is therefore $O(|\mathcal{S}|^3(|\mathcal{S}_2|+|\mathcal{Z}|)^2)$

During the downward pass, in a leader's node n we find for each $p \in P_n$ a line segment in S_n^2 on which it lies. This therefore takes $O(\left|S_n^1\right|)$ operations. For the two ending points of this line, we traverse S_v^2 for every child v of n. This therefore takes $O(|\mathcal{S}|(|\mathcal{S}_2|+|\mathcal{Z}|))$. Because the number of possible p's is $O(|\mathcal{S}_2|+|\mathcal{Z}|)$, processing the leader's node takes $O(|\mathcal{S}|(|\mathcal{S}_2|+|\mathcal{Z}|)^2$. In a follower's node n, we simply traverse all line segments of n's children. This therefore takes $O(|\mathcal{S}||\mathcal{S}_2|(|\mathcal{S}_2|+|\mathcal{Z}|)^2)$. Because we do these operations in every node, the total time required is again in $O(|\mathcal{S}|^3(|\mathcal{S}_2|+|\mathcal{Z}|)^2)$, which is therefore the total runtime of the algorithm. □

3.2 No chance, General graph

Now, let us relax the requirement that the graph representing the extensive form game should be acyclic, while keeping the property that the players receive utility only after they reach a leaf. First, we examine the memory needed to represent optimal strategies. We show that the memory requirements do not change from the DAG case, as we are able to use exactly the same arguments. However, it might still be the case that the optimal strategy would consist of a cycle through the memory states. Therefore, we will show that in fact there are no infinite strategies, because in the optimal strategy no point in the utility space is visited twice.

Theorem 3.3. In a stochastic game on a graph without chance nodes, there is a positionally determined strategy without cycles realizing the leader's maxmin value.

Proof: The maxmin strategy is pure and positionally determined by the same argument as in Lemma 3.1. Suppose that the strategy containts a cycle, visiting v at least twice in the history $\pi = n_0a_0 \dots n_{i-1}a_{i-1}va_v^1n_ia_i \dots n_{j-1}a_{j-1}va_v^2 \dots$. It is a pure strategy, so we can skip the part between n_i and n_{j-1} and go to n_j directly. □

Theorem 3.4. For every Stackelberg equilibrium σ in a stochastic game on a graph without chance nodes there is a strategy with memory $(\sigma', M, \mathcal{M}, m_0)$ such that $u(\sigma) = u(\sigma')$ and $|M| \leq |\mathcal{S}_2| + |\mathcal{Z}| + 1$.

Proof: The proofs of Theorem 3.1 and all lemmas before it considered only paths from root to some node. Therefore, the proof is still valid even on general graphs. □

While we have a limited number of memory states (and corresponding points in the utility space), to obtain finite strategies we need to prove that it is not the the case that some state is visited infinitely often. To do that, we first extend Lemma 3.3. To visualize this, imagine several Figures 3.1 chained after each other.

Lemma 3.8. Let σ be a Stackelberg equilbrium in a stochastic game on a graph without chance nodes. Let there be a path $\pi = v_0a_0 \dots v_ja_jv_{j+1} \dots v_k$ such that the leader randomizes in v_j between actions a_j and c and in v_k he randomizes between actions a

and b. Let $u_1^\sigma(\pi[k]a) \leq u_1^\sigma(\pi[k]b)$ and $u_2^\sigma(\pi[k]a) > u_2^\sigma(\pi[k]b)$. Then

$$slope(u^\sigma(\pi[k]a), u^\sigma(\pi[k]b)) \leq slope(u^\sigma(\pi[k]), u^\sigma(\pi[i]c)).$$

Proof: We will prove this using induction using Lemma 3.3. Let $i_1, \ldots, i_l$ be indices such that the leader randomizes after every $\pi[i_m]$ between actions a_{i_m} and b_m; and for all $1 \leq m < l$ $\sigma(\pi[i_m + 1{:}i_{m+1}]) = 1$. We continue via induction along m. For the base case, we have

$$slope(u^\sigma(\pi[i_1]a_{i_1}), u^\sigma(\pi[i_1]b_1)) \leq slope(u^\sigma(\pi[i_2]a_{i_2}), u^\sigma(\pi[i_2]b_2))$$

from Lemma 3.3. For the induction step, suppose we know that

$$slope(u^\sigma(\pi[i_1]a_{i_1}), u^\sigma(\pi[i_1]b_1)) \leq slope(u^\sigma(\pi[i_m]a_{i_m}), u^\sigma(\pi[i_m]b_m)).$$

From Lemma 3.3 we obtain

$$slope(u^\sigma(\pi[i_m]a_{i_m}), u^\sigma(\pi[i_m]b_m)) \leq slope(u^\sigma(\pi[i_{m+1}]a_{i_{m+1}}), u^\sigma(\pi[i_{m+1}]b_{m+1})),$$

which gives us

$$slope(u^\sigma(\pi[i_1]a_{i_1}), u^\sigma(\pi[i_1]b_1)) \leq slope(u^\sigma(\pi[i_{m+1}]a_{i_{m+1}}), u^\sigma(\pi[i_{m+1}]b_{m+1})).$$

□

By our limitation to strategies where the inequality from Lemma 3.3 is strict, we obtain $slope(u^\sigma(\pi[k]a), u^\sigma(\pi[k]b)) < slope(u^\sigma(\pi[k]), u^\sigma(\pi[i]c))$.

Theorem 3.5. For every Stackelberg equilibrium σ in an extensive form game on a graph there is a finite Stackelberg equilibrium σ' with equal expected utilities for both players in which for every path π from root to leaf with $\sigma(\pi) > 0$ no state is visited more than $|\mathcal{S}_2| + 1$ times.

Proof: Suppose we have a Stackelberg equilibrium σ such that there is a history $\pi = v_0 a_0 \ldots v_k$ such that the leader randomizes at least $|\mathcal{S}_2| + 1$ times in some $\pi[i]$. According to Lemma 3.7, each node in which the leader randomizes has a corresponding node of the follower, and strategies with the same corresponding noder have the same expected utility for both players. Because it is possible to mix more than $|\mathcal{S}_2|$ times, there has to be a followers node u such that two leader's nodes v_a, v_b on π correspond to u. But again, due to Lemma 3.7, we have $u^\sigma(\pi[a]) = u^\sigma(\pi[b])$, and due to Lemma 3.5 the strategies after $\pi[a]$ and $\pi[b]$ are the same – the leader randomizes between the same actions f and g with the same coefficient λ. But from Lemma 3.8 we have

$$slope(u^\sigma(\pi[a]f), u^\sigma(\pi[a]g)) < slope(u^\sigma(\pi[b]f), u^\sigma(\pi[b]g)$$

which contradicts that the strategies after $\pi[a]$ and $\pi[b]$ are the same. Therefore, after playing $|\mathcal{S}_2|$ mixtures there has to be a pure strategy going to leaf. Each state can therefore be visited at most $|\mathcal{S}_2| + 1$ times. □

As this theorem limits the depth of optimal strategies, it could be used in the future to devise an algorithm for computing equilibria on general graphs.

3.3 Chance, DAG

If we allow randomness to play a role in the game in the form of chance nodes, the problem of finding equilibrium becomes harder. Specifically, even on trees the problem of finding a Stackelberg equilibrium is NP-hard [7], and as trees are a subgroup of directed acyclic graphs, we cannot hope any better. However, our interest lies in the memory requirements. With the addition of nature states, we lose the ability to nicely trace the strategy via a string of pure strategies and pairs of follower's nodes and mixtures caused by them, as it was before (see Lemma 3.5). However, we can still show that the memory requirements do not grow. To do so, we once again establish that to each mixture, there is a follower's node that enforces this mixture and for each follower's node, we can consider only one strategy if we play to reach follower's maxmin value. This will allow us to break the whole game graph into pieces we can reason about, and in which the memory update scheme can be explained.

Lemma 3.9. Let σ be a Stackelberg equilibrium in a game in extensive form on a DAG with chance nodes. Let $\pi = v_0 a_0 \dots v_k$ be a history such that the leader mixes between actions a_k and b_k with probabilities α and $1-\alpha$ after π. Then either $u_1^\sigma(\pi a_k) \geq u_1^\sigma(\pi b_k)$ and $u_2^\sigma(\pi a_k) < u_2^\sigma(\pi b_k)$, or $u_1^\sigma(\pi a_k) \leq u_1^\sigma(\pi b_k)$ and $u_2^\sigma(\pi a_k) > u_2^\sigma(\pi b_k)$.

Proof: The proof from Lemma 3.2 holds also here, as chance nodes make convex combinations of utilities of their children, and convex combinations preserve inequalities. □

Lemma 3.10. In a two player game in extensive form on a DAG with chance nodes, there is a pure and positionally determined strategy σ such that for every node n $u_2^\sigma(n) = \mu(n)$.

Proof: Again, due to the fact that chance nodes make convex combinations only, the proof from Lemma 3.1 holds. □

Lemma 3.11. Let σ be a Stackelberg equilibrium in an extensive form game on a DAG with chance nodes. Let π_1, π_2 be two histories ending in a follower's node n such that $u_2^\sigma(\pi_1) = u_2^\sigma(\pi_2) = \mu(n)$. Then $u_1^\sigma(\pi_1) = u_1^\sigma(\pi_2)$.

Proof: The proof of Lemma 3.7 concerned only swapping probabilities to obtain a dominating strategy, therefore it holds also here. □

This allows us again to limit ourselves to a single continuing strategy whenever we reach a follower's node n via a history π such that $\mu(n) = u_2^\sigma(\pi)$. This splits the strategy into parts we can reason about, and will eventually provide us with a way how to update memory such that the strategy remains optimal. We now consider the opposite problem - we examine a situation in which the leader randomizes, and show there has to be a follower's node that is the cause of this randomization.

Lemma 3.12. Let σ be a Stackelberg equilibrium in an extensive form game on a DAG with chance nodes. Let $\pi = v_0 a_0 \dots v_k$ be a history played with nonzero probability such that the leader randomizes between actions a and b with probabilities α and $1-\alpha$ after π. Then there exists a $j < k$ such that the follower plays in v_j and $u_2^\sigma(\pi[j]) = \mu(v_j)$.

Proof: Suppose that for all $j < k$ such that the follower plays in v_j it holds that $u_2^\sigma(n) \neq \mu(n)$. By Lemma 3.9, let $u_2^\sigma(\pi a) < u_2^\sigma(\pi b)$ and $u_1^\sigma(a) \geq u_1^\sigma(b)$. If for any j $u_2^\sigma(\pi[j]) < \mu(v_j)$, then the follower has better utility if he plays another action in v_j, and σ is not an equilibrium. Hence, for all j we have $u_2^\sigma(\pi[j]) > \mu(v_j)$. However, then there has to exist an $\epsilon > 0$ such that the leader can play a after π with probability

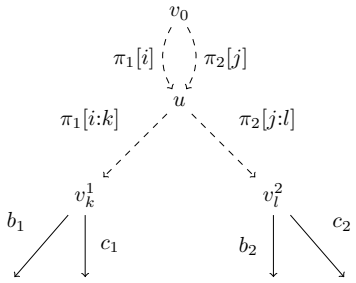

Figure 3.9. A situation from Lemma 3.9. Dashed lines represent paths with nonzero probability, straigh lines represent actions.

$\alpha + \epsilon$, because for each j we have

$$\begin{aligned} u_2^\sigma(\pi[j]) &= \sigma(\pi[j{:}k])u_2^\sigma(\pi[k]) + (1 - \sigma(\pi[j{:}k]))R_j \\ &= \sigma(\pi[j{:}k])(\alpha u_2^\sigma(\pi[k]a) + (1-\alpha)u_2^\sigma(\pi[k]b)) \\ &> \mu(v_j), \end{aligned}$$

where R_j is the utility obtain from all other continuations of $\pi[j]$ except for π. But by doing so, the utility of the leader can only increase, because $u_1^\sigma(a) \geq u_1^\sigma(b)$. Thus, σ is not optimal. □

As before, we turn our focus to limiting the number of randomizations the leader does. Due to the introduction of chance nodes we loose the nice geometric view on the problem, and have to turn to inequalities. To see the situation we are trying to avoid, refer to Figure 3.9.

Lemma 3.13. Let σ be a Stackelberg equilibrium in an extensive form game G on a DAG with chance nodes. Let $\pi_1 = v_0^1 a_0^1 \dots v_k^1$ and $\pi_2 = v_0^2 a_0^2 \dots v_l^2$ be two histories such that there is a follower's node u such that $u = v_i^1 = v_j^2$, i is the largest index for which $u_2^\sigma(\pi_1[i]) = \mu(u)$, j is the largest index for which $u_2^\sigma(\pi_2[j]) = \mu(u)$. Let $v_l^2 \neq v_m^1$ for all $m < k$ and $v_k^1 \neq v_n^2$ for all $n < l$. Then there is an equilibrium σ' with $u_1^\sigma(G) = u_1^{\sigma'}(G)$ in which the leader randomizes at most after one of π_1 and π_2.

Proof: Let $\pi_1 = v_0^1 a_0^1 \dots v_k^1$ and $\pi_2 = v_0^2 a_0^2 \dots v_l^2$ be two histories such that there is a follower's node u such that $u = v_i^1 = v_j^2$, i is the largest index for which $u_2^\sigma(\pi_1[i]) = \mu(u)$, j is the largest index for which $u_2^\sigma(\pi_2[j]) = \mu(u)$. Let $v_l^2 \neq v_m^1$ for all $m < k$ and $v_k^1 \neq v_n^2$ for all $n < l$. If the leader does not randomize after both π_1 and π_2, we can set $\sigma' = \sigma$. Therefore, suppose that after π_1 the leader randomizes between actions b_1 and c_1 with probabilities α and $1 - \alpha$, and after π_2 between actions b_2 and c_2 with probabilities β and $1 - \beta$, as shown in Figure 3.9. Without loss of generality, let

$$slope(u^\sigma(\pi_1 b_1), u^\sigma(\pi_1 c_1)) \leq slope(u^\sigma(\pi_2 b_2), u^\sigma(\pi_2 c_2). \tag{1}$$

By Lemma 3.9, let $u_2^\sigma(\pi_1 b_1) < u_2^\sigma(\pi_1 c_1)$, $u_1^\sigma(\pi_1 b_1) \geq u_1^\sigma(\pi_1 c_1)$, $u_2^\sigma(\pi_2 b_2) < u_2^\sigma(\pi_2 c_2)$ and $u_1^\sigma(\pi_2 b_2) \geq u_1^\sigma(\pi_2 c_2)$.

Because of Lemma 3.12, we know that both players play the same strategy after $\pi_1[i]$ and $\pi_2[j]$, that is $\sigma(\pi_1[i] \downarrow) = \sigma(\pi_2[j] \downarrow)$. This means that $u^\sigma(\pi_1[i]\pi_2[j{:}l]) = u^\sigma(\pi_2)$. Because also for all $m < k$ $v_l^2 \neq v_m^1$ and for all $n < l$ $v_k^1 \neq v_n^2$, by changing $\sigma(\pi[k])$ we do not affect $\sigma(\pi[l])$, and vice versa. Moreover, we can express $u^\sigma(\pi_1[i])$ as

$$u_i^\sigma(\pi_1[i]) = \sigma(\pi_1[i{:}k])u^\sigma(\pi_1) + \sigma(\pi_2[j{:}l])u^\sigma(\pi_1) + (1 - \pi_1[i{:}k] - \pi_2[j{:}l])R^i,$$

where R^i is the utility of player i obtained in all other continuations of $\pi_1[i]$ except for π_1 and $\pi_1[i]\pi_2[j{:}l]$. Hence, we aim to change σ, creating a strategy σ', such that the

follower receives the same utility in $\pi_1[i]$ and $\pi_2[j]$ as before, while the leader's utility becomes greater or equal to the old one.

As in Lemma 3.12, we know that for all $i < m < k$ such that the follower plays in v_m^1 $u_2^\sigma(\pi_1[m]) > \mu(v_m^1)$ and for all $j < n < l$ such that the follower plays in v_n^2 $u_2^\sigma(\pi_2[n]) > \mu(v_n^2)$. Again, we have

$$\begin{aligned} u_2^\sigma(\pi_1[m]) &= \sigma(\pi_1[m{:}k])u_2^\sigma(\pi_1[k]) + (1 - \sigma(\pi_1[m{:}k]))R_m^2 \\ &= \sigma(\pi_1[m{:}k])(\alpha u_2^\sigma(\pi_1[k]b_1) + (1-\alpha)u_2^\sigma(\pi_1[k]c_1)) + (1 - \sigma(\pi_1[m{:}k]))R_m^2 \\ &> \mu(v_m^1). \end{aligned}$$

Hence, there has to exists an $\epsilon > 0$ we can add to α without breaking any of the constraints that $u_2^{\sigma'}(\pi_1[m]) \geq \mu(v_m)$. Also, subtracting some δ from β cannot invalidate these constraints, because by doing so the follower's utility increases. Therefore, let $\epsilon' > 0$ be the maximum number such that $\alpha + \epsilon' \leq 1$ for which there exists a δ' such that $\beta - \delta' \geq 0$, for all $i < m < k$ it holds that

$$\sigma(\pi_1[m{:}k])((\alpha+\epsilon')u_2^\sigma(\pi_1 b_1) + (1-\alpha-\epsilon')u_2^\sigma(\pi_1 c_1)) + (1-\sigma(\pi_1[m{:}k]))R_m^1 \geq \mu(v_m^1) \quad (2)$$

and the utility in u remains unchanged, which requires

$$-\sigma(\pi[i{:}k])\epsilon'(u_2^\sigma(\pi_1 b_1) - u_2^\sigma(\pi_1 c_1)) = \sigma(\pi[j{:}l])\delta'(u_2^\sigma(\pi_2 c_2) - u_2^\sigma(\pi_2 b_2)). \quad (3)$$

Because no condition on ϵ' is a strict inequality, we are sure that ϵ' is defined correctly.

We can now create σ' by copying σ everywhere except after π_1 and π_2, and setting $\sigma'(\pi_1, b_1) = \alpha + \epsilon'$, $\sigma'(\pi_1, c_1) = 1 - \alpha - \epsilon'$, $\sigma'(\pi_2, b_2) = \beta - \delta'$ and $\sigma'(\pi_2, c_2) = 1 - \beta + \delta'$. Because of the definition of ϵ' and δ', we have $0 \leq \alpha + \epsilon' \leq 1$ and $0 \leq \beta - \delta' \leq 1$, and σ' is defined correctly. Because of constraint (2), we have for all $i < m < k$

$$\begin{aligned} u_2^{\sigma'}(\pi_1[m]) &= \sigma(\pi_1[m{:}k])((\alpha+\epsilon')u_2^\sigma(\pi_1[k]b_1) \\ &\quad + (1-\alpha-\epsilon')u_2^\sigma(\pi_1[k]c_1)) + (1-\sigma(\pi_1[m{:}k]))R_m^1 \\ &\geq \mu(v_m^1) \end{aligned}$$

and the follower does not deviate in any such m. Because $u_2^\sigma(\pi_2 b_2) < u_2^\sigma(\pi_2 c_2)$, we have

$$\begin{aligned} u_2^{\sigma'}(\pi_2) &= (\beta - \delta')u_2^\sigma(\pi_2 b_2) + (1 - \beta + \delta')u_2^\sigma(\pi_2 c2) \\ &= u_2^\sigma(\pi_2) + \delta'(u_2^\sigma(\pi_2 c2) - u_2^\sigma(\pi_2 b_2)) \\ &> u_2^\sigma(\pi_2). \end{aligned}$$

Hence, $u_2^{\sigma'}(\pi_2)$ is strictly better for the follower and he does not deviate.

Because we set ϵ' to be the maximum ϵ such that the constraints hold, then either

- $\alpha + \epsilon' = 1$. In such case $\sigma'(\pi_1, b_1) = 1$ and the leader does not randomize after π_1.
- $\beta - \delta' = 0$. Then $\sigma'(\pi_2, c_2) = 1$ and the leader odes not randomize after π_2.
- $u_2^{\sigma'}(\pi_1[m]) = \mu(v_m^1)$ for some $i < m < k$.

Therefore, either the leader does not mix after one of π_1, π_2, or i is not the largest index for which $u_2^{\sigma'}(\pi_1[i]) = \mu(v_i^1)$. We now just have to show that the expected utilities of σ' are not lower than those of σ.

Let us now examine the utilities in u for both players. For the follower, we have

$$\begin{aligned}
u_2^{\sigma'}(\pi_1[i]) &= \sigma(\pi_1[i{:}k])u_2\sigma'(\pi_1) + \sigma(\pi_2[j{:}l])u_2\sigma'(\pi_2) + (1 - \sigma(\pi_1[i{:}k]) - \sigma(\pi_2[j{:}l]))R_i^2 \\
&= \sigma(\pi_1[i{:}k])((\alpha+\epsilon')u_2^\sigma(\pi_1 b_1) + (1-\alpha-\epsilon')u_2^\sigma(\pi_1 c_1)) \\
&+ \sigma(\pi_2[j{:}l])((\beta-\delta')u_2^\sigma(\pi_2 b_2) + (1-\beta+\delta')u_2^\sigma(\pi_2 c_2)) + (1 - \sigma(\pi_1[i{:}k]) - \sigma(\pi_2[j{:}l]))R_i^2 \\
&= \sigma(\pi_1[i{:}k])u_2^\sigma(\pi_1) + \sigma(\pi_1[i{:}k])\epsilon'(u_2^\sigma(\pi_1 b_1) - u_2^\sigma(\pi_1 c_1)) \\
&+ \sigma(\pi_2[j{:}l])u_2\sigma(\pi_2) + \sigma(\pi_2[j{:}l])\delta'(u_2^\sigma(\pi_2 c_2) - u_2^\sigma(\pi_2 b_2)) + (1 - \sigma(\pi_1[i{:}k]) - \sigma(\pi_2[j{:}l]))R_i^2 \\
&= \sigma(\pi_1[i{:}k])u_2^\sigma(\pi_1) + \sigma(\pi_2[j{:}l])u_2\sigma(\pi_2) + (1 - \sigma(\pi_1[i{:}k]) - \sigma(\pi_2[j{:}l]))R_i^2 \\
&= u_2^\sigma(\pi_1[i]),
\end{aligned}$$

where R_i^2 is the follower's utility from all continuations of $\pi_1[i]$ except for $\pi_1[i{:}l]$ and $\pi_1[i]\pi_2[j{:}l]$. Hence, the follower's utility remains the same.

As for the leader, from the condition (1) on slopes of mixtures and equation (3) we obtain

$$-\sigma(\pi[i{:}k])\epsilon'(u_1^\sigma(\pi_1 b_1) - u_1^\sigma(\pi_1 c_1)) \leq \sigma(\pi[j{:}l])\delta'(u_1^\sigma(\pi_2 c_2) - u_1^\sigma(\pi_2 b_2)).$$

Hence, we obtain

$$\begin{aligned}
u_1^{\sigma'}(\pi_1[i]) &= \sigma(\pi_1[i{:}k])u_1\sigma'(\pi_1) + \sigma(\pi_2[j{:}l])u_1\sigma'(\pi_2) + (1 - \sigma(\pi_1[i{:}k]) - \sigma(\pi_2[j{:}l]))R_i^1 \\
&= \sigma(\pi_1[i{:}k])((\alpha+\epsilon')u_2^\sigma(\pi_1 b_1) + (1-\alpha-\epsilon')u_2^\sigma(\pi_1 c_1)) \\
&+ \sigma(\pi_2[j{:}l])((\beta-\delta')u_2^\sigma(\pi_2 b_2) + (1-\beta+\delta')u_2^\sigma(\pi_2 c_2)) + (1 - \sigma(\pi_1[i{:}k]) - \sigma(\pi_2[j{:}l]))R_i^1 \\
&= \sigma(\pi_1[i{:}k])u_2^\sigma(\pi_1) + \sigma(\pi_1[i{:}k])\epsilon'(u_2^\sigma(\pi_1 b_1) - u_2^\sigma(\pi_1 c_1)) + \sigma(\pi_2[j{:}l])u_2\sigma(\pi_2) \\
&+ \sigma(\pi_2[j{:}l])\delta'(u_2^\sigma(\pi_2 c_2) - u_2^\sigma(\pi_2 b_2)) + (1 - \sigma(\pi_1[i{:}k]) - \sigma(\pi_2[j{:}l]))R_i^1 \\
&\geq \sigma(\pi_1[i{:}k])u_2^\sigma(\pi_1) + \sigma(\pi_2[j{:}l])u_2\sigma(\pi_2) + (1 - \sigma(\pi_1[i{:}k]) - \sigma(\pi_2[j{:}l]))R_i \\
&= u_2^\sigma(\pi_1[i]),
\end{aligned}$$

where R_i^1 is the leader's utility from all continuations of $\pi_1[i]$ except for $\pi_1[i{:}l]$ and $\pi_1[i]\pi_2[j{:}l]$. We have found σ' with $u_1^{\sigma'}(G) \geq u_1^\sigma(G)$ and $u_2^{\sigma'}(G) = u_2^\sigma(G)$ in which the leader randomizes after at most one of π_1, π_2. □

Note that the conditions of this lemma allow for $v_k^1 = v_l^2$. This means that after passing a node n via any history π with $u_2^\sigma(\pi) = \mu(n)$, the leader randomizes at most once per leader's node v, provided the path between n and v doesn't contain a follower's node u for which the strategy would yield utility $\mu(u)$. This opens up a way to construct strategies with memory. To do so, let us first prove a helpful lemma.

Lemma 3.14. Let σ be a Stackelberg equilibrium in an extensive form game on a DAG with chance nodes. For each $v \in \mathcal{S}$ let $U_v = \{u^\sigma(\pi) | \pi = v_0 a_0 \dots v_k a_k v, \sigma(\pi) > 0\}$ be a set of utility points attainable at v. Then $|U_v| \leq |\mathcal{S}_2|\,(|\mathcal{S}_1| + 1)$.

Proof: Suppose for contradiction that for a node v we have

$$|U_v| \geq |\mathcal{S}_2|\,(|\mathcal{S}_1| + 1) + 1.$$

Let us partition the paths leading to v according to the last follower's node for which the follower's maxmin value is obtained. That is, for every $n \in \mathcal{S}_2$ let $\pi = v_0 a_0 \dots v_k a_k v$ belong to P_n iff

- there is an $i < k$ such that $n = v_i$,
- $\sigma(\pi) > 0$,
- $u_2^\sigma(\pi[i]) = \mu(n)$,

- for each $i < j < k$ it holds that $u_2^\sigma(\pi[i]) \neq \mu(v_j)$

We can partition U_v according to individual P_n as

$$U_v^n = \{u^\sigma(\pi) | \pi \in P_n\}\,.$$

Due to definition of P_n, we know that all U_v^n partition U_v. Because $|U_v| \geq |\mathcal{S}_2|\,|\mathcal{S}_1| + 1$, there has to exist an $n \in \mathcal{S}_2$ such that $|U_v^n| \geq |\mathcal{S}_1| + 2$. Because we are in a Stackelberg equilibrium, the utility points reached can be differentiated only by different commitments of the leader (the follower always best responds). But because we have at least $|\mathcal{S}_1|+2$ utility points, there have to be at least two pairs of utility points u^1, u^2 and v^1, v^2 such that the commitments made to obtain u^1 and u^2 differ in some leader's node s_1 and the commitments made to obtain v^1 and v^2 differ in some leader's node s_2. Let π_1^u be the path which reaches the node s_1 when playing to achieve utility u^1. Similarly, define $\pi_2^u, \pi_1^v, \pi_2^v$. On all these paths, node s_1 and s_2 must be encountered before going through a follower's node n with the utility equal to $\mu(n)$, because after passing such a node the strategies coincide due to Lemma 3.11. According to Lemma 3.13, the leader mixes in at most one of $\pi_1^u, \pi_2^u, \pi_1^v, \pi_2^v$. Without loss of generality, let π_1^u, π_2^u be the pair of histories in whitch the leader does not mix. But we could change the strategies as in Lemma 3.13 to obtain a dominating strategy, and σ is not an equilibrium. □

Theorem 3.6. Let G be an extensive form game on a DAG with chance nodes. For every Stackelberg equilibrium $\sigma : \Pi \times \mathcal{A} \to [0,1]$ there is an equilibrium with memory $\sigma' : M \times (\mathcal{S} \times \mathcal{A}) \to [0,1]$ such that $u^\sigma(G) = u^{\sigma'}(G)$ and $|M| \leq |\mathcal{S}_2|\,(|\mathcal{S}_2| + |\mathcal{Z}|) + 1$.

Proof: Let $\sigma : \Pi \times \mathcal{A} \to [0,1]$ be a Stackelberg equilibrium in G. Let $v_0, \ldots v_{|\mathcal{S}|}$ be a topologic order of nodes in $\mathcal{S}$. We will traverse the nodes in this ordering, creating the equilibrium with memory σ', set of memory states M and the memory update function $\mathcal{M}$. For every v_i we will create a number r_i (the number of memory states in which v_i can be reached) and a set R_i of tuples (m, u_1, u_2), where m is a memory state and u_1, u_2 are the utilities of leader and the folower which should be obtained when playing with memory state m.

Initially, let $r_0 = 1$ and $R_0 = \{(0, u_1^\sigma(G), u_2^\sigma(G))\}$. For all other nodes set $r_i = 0$ and $R_i = \emptyset$. In a node v_i, do the following for each $(m, u_1, u_2) \in R_i$. Take any history $\pi = w_0 a_0 \ldots w_k a_k v_i$ with $u^\sigma(\pi) = (u_1, u_2)$ and $\sigma(\pi) > 0$. Let $\sigma'(m; v_i, a) = \sigma(\pi, a)$ for all actions a available in v_i. For each action a available in v_i let n_a be the node reached by playing a in v_i. If $\sigma(\pi, a) = 0$, we do not need to define a memory transition as a will never be played. (An exception to this are follower's nodes, in which we need to setup threats. We will deal with these later.) Otherwise, there are two options. If there is a memory state m' such that $(m', u_1^\sigma(\pi a), u_2^\sigma(\pi a)) \in R_{n_a}$, set $\mathcal{M}(m; v_i, a) = m'$. Otherwise, set $\mathcal{M}(m; v_i, a) = r_i$, add $(r_i, u_1^\sigma(\pi a), u_2^\sigma(\pi a))$ to R_i and increment r_i by one. Finally, after $v_{|\mathcal{S}|}$ is processed, set $M = \{0, 1, \ldots, \max_i \{r_i\}\}$.

Now we just have to add one more memory state to signify that the follower's maxmin value should be reached if the follower deviates. Let this state be $m^* = \max_i \{r_i\} + 1$. Therefore, for every follower's node v_j and for every $0 \leq m \leq r_j$ for each action a available in v_j such that $\sigma'(m; v_j, a) = 0$ set $\mathcal{M}(m; v_j, a) = m^*$. To setup the maxmin strategy, let $\sigma'(m^*; v, a) = \sigma_\mu(v_j, a)$ for every $v \in \mathcal{S}$ and $\mathcal{M}(m^*; v, a) = m^*$. Finally, add m^* to M.

To see that σ' is defined correctly, we have to show that for every v_i and every $(m, u_1, u_2) \in R_i$ there is a history π ending in v_i such that $u^\sigma(\pi) = (u_1, u_2)$ and $\sigma(\pi) > 0$. But this is trivially true due to construction of R_i.

We now just have to show that $|M| \leq |\mathcal{S}_2|\,(|\mathcal{S}_2| + |\mathcal{Z}|) + 1$. Suppose for contradiction that $|M| > |\mathcal{S}_2|\,(|\mathcal{S}_2| + |\mathcal{Z}|) + 1$. This means there is a node with $r_i \geq |\mathcal{S}_2|\,(|\mathcal{S}_2| + |\mathcal{Z}|) + 1$

(This shift by one is caused by the one extra memory state m^*). But this means that $|R_i| \geq |\mathcal{S}_2|\,(|\mathcal{S}_2| + |\mathcal{Z}|) + 1$, and since a tuple (m, u_1, u_2) is added to R_i only when it does not contain another record (m', u_1, u_2), we obtain for the set U_i of utility points attainable at i that $|U_i| = |R_i| \geq |\mathcal{S}_2|\,(|\mathcal{S}_2| + |\mathcal{Z}|) + 1$, which contradicts Lemma 3.14. □

Let us now for completeness state the complexity of finding Stackelberg equilibria with memory. Because finding an equilibrium on trees is NP-hard [7] and finding an equilibrium on trees is a subproblem of finding an equilibrium on DAGs (as an equilibrium on trees is always memoryless), we obtain NP-hardness immediately.

Theorem 3.7. Finding a Stackelberg equilibrium with memory in an extensive form game on a DAG with chance nodes is NP-hard.

3.3.1 Approximating strategies

Due to NP-hardness of finding optimal strategies with memory we turn to approximating these strategies. Our interest will be in additive approximation - for a game with utilities in the interval $[0, 1]$ we try to find a strategy such that the expected leader's utility obtain by following this strategy differs from the optimal strategy by at most ϵ. We adapt the algorithm from [1] to work on DAGs. The upward dynamic programming pass remains exactly the same. During downward pass, when the strategy is retrieved, some additional work has to be done to retrieve the strategy with memory. The downward pass is closely related to the proof of Theorem 3.6.

Theorem 3.8. In an extensive form game G on a binary DAG with chance nodes with utilities in $[0, 1]$, a strategy profile σ with memory can be found for which $u_1^{\sigma}(G) \geq u_1^{\sigma^*}(G) - \epsilon$, where σ^* is a Stackelberg equilibrium. This can be done in $O(\epsilon^{-3}(H_G)^3\,|\mathcal{S}| + |\mathcal{S}|\,|\mathcal{S}_2|\,|\mathcal{S}_1|)$, where H_G is the height of the DAG.

Proof: The algorithm consists of two upward passes and a downward pass. During the first upward pass, the follower's maxmin value $\mu(n)$ is computed for every node. During the second pass, a table of utilities is constructed for every node, and a downward pass, during which the strategy is retrieved. Let us summarize how the upward pass works. First, the utilities are scaled by $((H_G + 1)/\epsilon)$. Let the highest scaled utility be U. Then, a table A_T is created for each subgraph of the game such that

- If $A_T[k] > -\infty$, the leader has a strategy for the subgraph that yields the follower utility $A_T[k]$ while the leader's utility is at least k.
- No strategy in the subgraph can offer the follower strictly more than $A_T[k]$ while yielding at least $k + H_T$ for the leader.
- The entries $A_T[k]$ are non-increasing and $A_T[U + 1] = -\infty$

To simplify notation, by A_v we mean A_T where T is the subgraph rooted by v.

Upward pass If v is a leaf with utilities (u_1, u_2), let

$$A_v[k] = \begin{cases} u_2 & \text{if } k \leq u_2, \\ -\infty & \text{otherwise.} \end{cases}$$

If v is a leader's node with children L and R played with probabilities p and $1 - p$, let

$$A_v[k] = \max_{i,j,p} \{pA_L[i] + (1-p)A_R[j] | pi + (1-p)j \geq k\}.$$

If v is a chance node, let

$$A_v[k] = \max_{i,j} \{pA_L[i] + (1-p)A_R[j] | pi + (1-p)j \geq k\}.$$

And finally, if v is a follower's node, let

$$A_v[k] = \max\left\{A_L[k]\downarrow_{\mu(R)}, A_R[k]\downarrow_{\mu(L)}\right\},$$

where

$$x\downarrow_\mu = \begin{cases} x & \text{if } x \geq \mu, \\ -\infty & \text{otherwise.} \end{cases}$$

This dynamic programming algorithm was originally stated for trees. However, when we consider a DAG and it's equivalent tree form, whenever a table is computed for two nodes v', v'' which maps onto a node v in a DAG, the subtrees under v' and v'' are identical, hence also the tables $A_{v'}, A_{v''}$ are identical. Therefore, we can compute it on the DAG directly.

Downward pass During the downward pass, we need to construct the strategy with memory $\sigma' : M \times (\mathcal{S} \times \mathcal{A}) \to [0,1]$, the memory set M (which will be a set of integers), the memory update function $\mathcal{M}$ and the initial state m_0. We proceed analogously to the proof of Theorem 3.6. For each node we create a set Q_i of pairs (m,i), meaning that in memory state m the strategy corresponding to i should be played. We use one special value for i - if $i = -1$, it singnifies that we are off the equilibrium path and follower's maxmin value should be achieved.

In the root node n the solution i^* of the game is $l^* = \max\{i | A_T[i] > -\infty\}$. Hence, we let $Q_n = (0, i^*)$. Set the initial state $m_0 = 0$. We now reason about individual types of nodes, dealing first with the maxmin strategy. For every $(m,k) \in Q_v$, do the following procedure.

In any node v, if $(m,-1) \in R_v$, the maxmin value for the leader should be played. Therefore, let $\sigma(m; v, L) = \sigma_\mu(v, L)$. If there is a state m' such that $(m', -1) \in R_L$, set $\mathcal{M}(m; v, L) = m'$. Otherwise, let $m' = |Q_L|$, set $\mathcal{M}(m; v, L) = m'$ and add $(m', -1)$ to Q_L. The strategy for R is defined analogously.

If v is a leader's node, let

$$i, j, p \in \operatorname*{argmax}_{i,j,p} \{pA_L[i] + (1-p)A_R[j] | pi + (1-p)j \geq k\}.$$

That is, i, j, p are the maximizing values responsible for the table entry $A_v[i]$. Let $\sigma(m; v, L) = p$. If there is a state m' such that $(m', i) \in Q_R$, set $\mathcal{M}(m; v, L) = m'$. Otherwise, let $m' = |Q_L|$, set $\mathcal{M}(m; v, L) = m'$ and add $(m', -1)$ to Q_L. The strategy for R is defined analogously.

In a chance node, we do not have to define a strategy. However, we still have to define the memory update function. Let therefore

$$i, j \in \operatorname*{argmax}_{i,j} \{pA_L[i] + (1-p)A_R[j] | pi + (1-p)j \geq k\}$$

and define $\mathcal{M}$ as in leader's node.

In a follower's node, the value in $A_T[k]$ either comes from the left child L, or the right child R. If the maximum comes from L, set $\sigma(m; v, L) = 1$. Again, if there is a state m' such that $(m', i) \in Q_L$, set $\mathcal{M}(m; v, L) = m'$. Otherwise, let $m' = |Q_L|$, set $\mathcal{M}(m; v, L) = m'$ and add $(m', -1)$ to Q_L. As for the right child, the maxmin value for the follower should be attained. Therefore, if there is an m' such that $(m', -1) \in Q_R$, set $\mathcal{M}(m; v, R) = m'$. Otherwise, let $m' = |Q_R|$, set $\mathcal{M}(m; v, R) = m'$ and add $(m', -1)$ to R_L. If the maximum comes from R, define the strategy analogously.

For each state v, the memory states in Q_v are integers from 0 to $|Q_v|$ due to the construction of Q_v. Therefore, we can set $M = \{0 \dots \max_v \{|Q_v|\}\}$.

Correctness The upward pass was proved correct in [1]. As for the downward pass, in the rood node we commit to the strategy which yields the solution. Whenever a pair (m, i) is added to Q_L (Q_R) in v, it is because i is a maximizer of some $A_v[k]$, and is a part of optimal strategy. Such a pair is not added only in the case that the corresponding index i already has some memory state m' such that $(m', i) \in Q_L$. In such a case, the memory state is changed to m' when going to L, and therefore yields correct utility.

Runtime The backwards induction runs in $O(|\mathcal{S}|)$ [1]. The upward pass runs in $O(\epsilon^{-3}(H_G)^3 |\mathcal{S}|)$. As for the downward pass, if we manage to bound the size of M, we simultaneously bound the size of every Q_v. But the memory bound $|\mathcal{S}_2| |states_1|$ can be obtained analogously to Theorem 3.6 and Lemma 3.14, except working with table indices. Because at most $|Q_v|$ pairs (m, i) are computed in every state, the downward pass takes $O(|\mathcal{S}| |\mathcal{S}_2| |\mathcal{S}_1|)$. The complexity of the whole algorithm is thus $O(\epsilon^{-3}(H_G)^3 |\mathcal{S}| + |\mathcal{S}| |\mathcal{S}_2| |\mathcal{S}_1|)$. □

Chapter 4
Experiments

We conducted experiments on randomly generated games to evaluate the scalability of the approximation algorithm from Theorem 3.8. To do so, we implemented a tool for generated random DAG games. Then, we conducted a series of experiments to determine how the algorithms behave. We have found that with our hardware we could solve DAGs with 500 nodes in under two minutes time with approximation factor $\epsilon = 0.05$, and around 45 minutes with approximation factor $\epsilon = 0.02$. We have also examined the number of nodes of the tree version of the DAG, and compared it with the output of our algorithm. Also, we have found out that in fact a very small number of memory state suffices to represent optimal strategies. All experiments were done in Python 3.5.

4.1 Generating random DAGs

To generate random DAGs we implemented a simple tool. Our tool takes two parameters, the number of desired nodes n and a leaf probability parameter p, which controlls the percentage of nodes of the DAG which are terminal. First, we create n nodes $v_0, \ldots, v_n$. Because we are generating a DAG, we assume that the nodes are already topologically ordered. Because we are interested in DAGs in which every node is reachable from v_0, we first traverse the nodes forwards, and generate a parent v_j for each node v_i such that $j < i$. We allow only parents with less than two children to be generated. Second, we traverse the nodes backwards, and for each node which has only one child we generate a second one (to ensure that our DAG is binary). And finally third, we traverse all nodes which have 0 children and for each such node we generate 2 children with probability $1 - p$. This means that such node remains a leaf with probability p.

After we generate a DAG, we assign each node a player with equal probability. For chance nodes, we generate probability q of playing the left action from uniform distribution, letting the probability of going right be $1 - q$. For leaves, we generate utilities of both players from the uniform distribution on $[0, 1]$. While we considered some more sophisticated method for generating utilities, where the utilities of the leader and follower would correlate, we decided not to implement such approach because it does not affect the actual runtime of the algorithm.

4.2 Scalability

We conducted a series of experiments on our randomly generated DAGS. We considered graphs with $n \in \{10, 50, 100, 200, 500\}$ nodes, with the leaf probability $p \in \{0.1, 0.5, 0.9\}$ and the approximation factors $\epsilon \in \{0.1, 0.05, 0.02\}$. For each combination of parameters, the algorithm was run 21 times.

In Figures 4.1, 4.2 and 4.3, there are the results of our scalability experiments. As we can see, the factor that influences the runtime of the algorithm the most is the

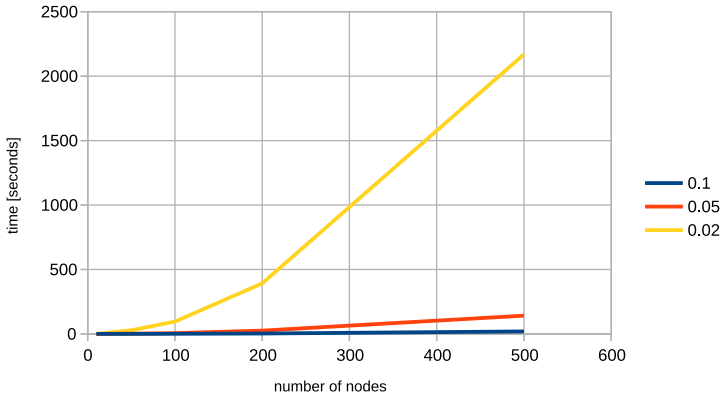

Figure 4.1. The average runtime of the ϵ aproximation algorithm on a DAG with leaf probability $p = 0.1$, for varying number of nodes and the approximation factor ϵ.

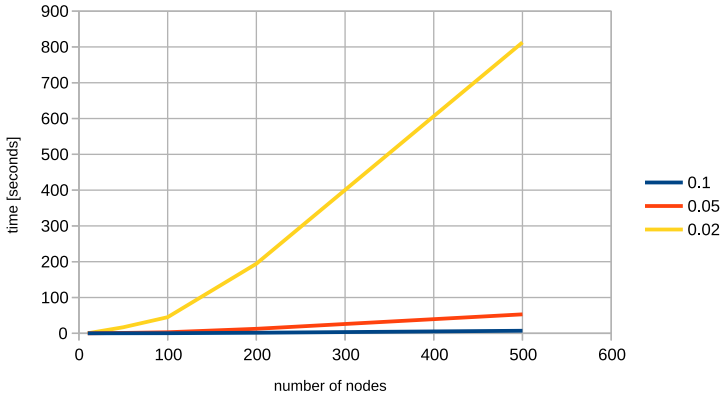

Figure 4.2. The average runtime of the ϵ aproximation algorithm on aDAG with leaf probability $p = 0.5$, for varying number of nodes and the approximation factor ϵ.

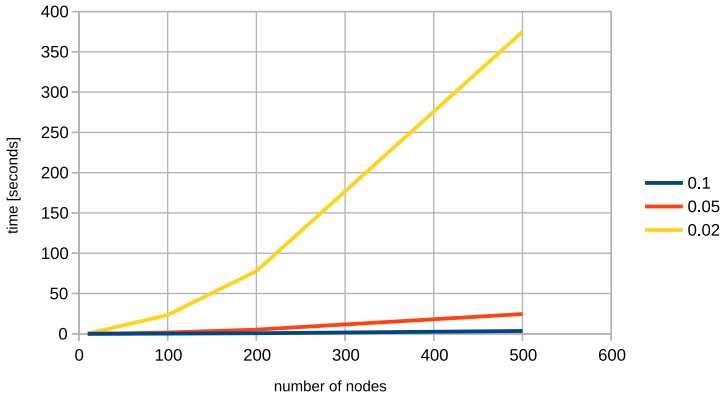

Figure 4.3. The average runtime of the ϵ aproximation algorithm on a DAG with leaf probability $p = 0.9$, for varying number of nodes and the approximation factor ϵ.

aproximation factor ϵ. This is however no surprise, because the complexity of the algorithm depends on ϵ^{-3}. While for approximation factor 0.1 we obtain $0.1^{-3} = 1000$. However, for approximation factor 0.02 we obtain a runtime three orders of magnitudes larger, $0.02^{-3} = 125000$.

p/n	10	50	100	200	500
0.1	0.96	0.33	0.17	0.13	0.04
0.5	0.97	0.65	0.48	0.45	0.35
0.9	0.92	0.70	0.95	0.80	0.94

Table 4.1. The average reduction of the number of remembered strategies. The members of this table are computed as the ration of the sum of numbers of memory states achievable in ever node to the number of the inner nodes in the tree version of a DAG.

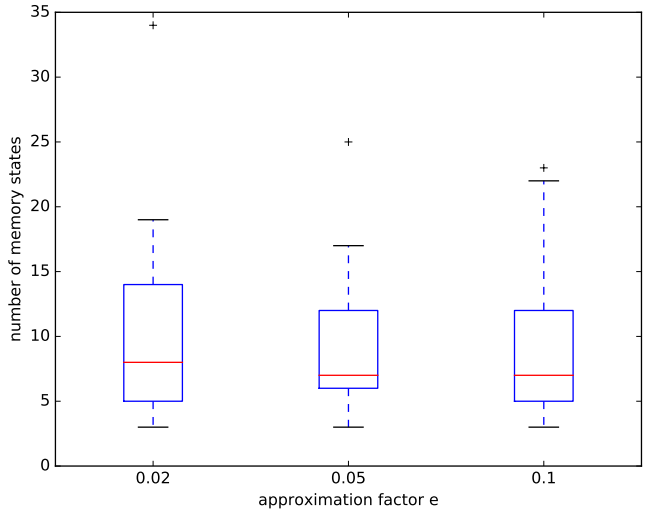

Table 4.2. The number of memory states in the ϵ approximation of a Stackelberg equilibrium for varying ϵ. All games have $n = 500$ nodes and the leaf probability $p = 0.1$.

The computation of the table A_T takes a significantly longer time for non-terminal nodes than for leaf nodes. Therefore, we would expect to see longer runtimes for graphs with lower ammount of leaf nodes, that is with low leaf probability p. And indeed, our results show that while the computation of an approximate strategy of games with 500 nodes and an approximation factor 0.02 took on average 374 seconds for games with leaf probability 0.9, for games with leaf probability 0.1, the computation of the approximation algorithm took on average 2168 seconds, almost five times as much.

4.2.1 Advantages of strategies with memory

To provide experimental evidence of usefulness of equilibria with memory, we collected some additional data about the strategies found by our algorithm. First, we examined the advantage obtained by using strategies with memory instead of transforming the DAG into a game tree and computing strategy there. We present the results of this investigation in Table 4.1. The numbers in the cells were obtained as follows: first, we computed the actual number of defined strategies, that is for every node we calculated in how many memory states is this node reachable. We added these number for every state, obtaining the total number of computed strategies. Then, we divided it by the number of inner nodes of the tree version of the game.

For games with a big probability that a given state is a leaf, the numbers vary around 0.9. This is caused by the fact that the tree version of the game is not much larger than the DAG. On the other hand, for games with a small number of leaf nodes there are many nodes with multiple parents, and therefore the tree transformation is very

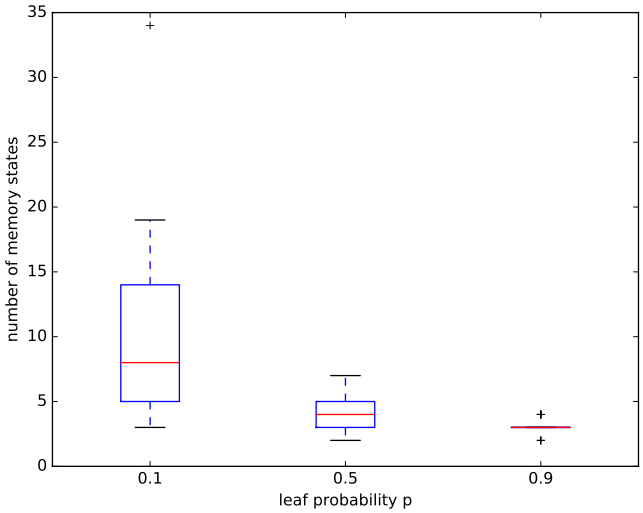

Table 4.3. The number of memory states in the ϵ approximation of a Stackelberg equilibrium for varying leaf probability p. All games have $n = 500$ nodes and the approximation factor is $\epsilon = 0.02$.

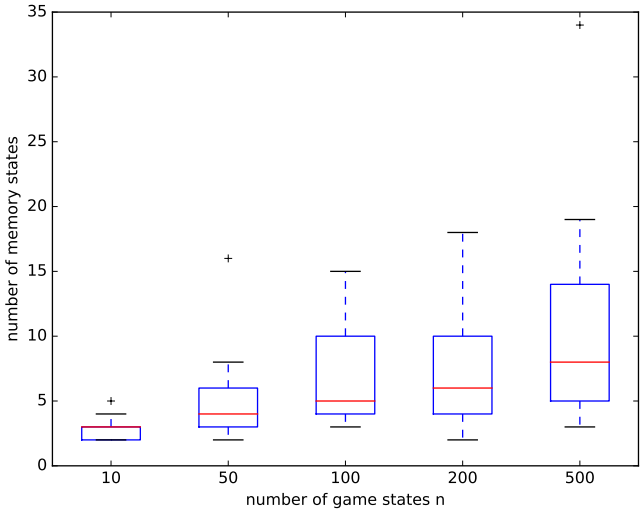

Table 4.4. The number of memory states in the ϵ approximation of a Stackelberg equilibrium for varying number of nodes n. All games have leaf probability $p = 0.1$ and the approximation factor is $\epsilon = 0.02$

large. Indeed, for games with 500 nodes the strategies with memory use only 4% of the memory states which would be needed for remembering the strategy on the tree.

Second, we turned our interest to the number of memory states in the whole game. The results of these experiments are presented in Figures 4.2, 4.3 and 4.4. In Figure 4.2 are the distributions of memory states for varying approximation factor. These distributions are very similar, suggesting that the number of memory states does not depend on the approximation factor. This agrees with our results.

In Figure 4.3 we examine how the number of memory states varies with the leaf probability. We find out that for games with a lot of internal nodes, that is games with low leaf probability, the memory requirements are signifficantly higher. Again, this

experimentally confirms our result that the number of used memory states depends on the number of internal nodes.

Finally, in Figure 4.4 we see that the average number of memory states grows approximately linearly with increasing number of states of the game. However, there are occasional outliers present. This grows suggests that the bound on the memory states could be perhaps lowered to a linear one, as our bound is quadratic.

Chapter 5
Conclusion

We have considered the memory requirements of representation of Stackelberg equilibria of different kinds of games. We have found out that for games on directed acyclic graphs without chance nodes the needed memory is linear in the size of the game. We have devised an algorithm which finds this equilibrium in polynomial time.

For stochastic games on graphs without chance nodes, we have shown that the memory bound is the same as with directed acyclic graphs. We have also shown that the Stackelberg equilibria in such games consist of paths with length bounded by the size of the game.

Fo games on directed acyclic graphs with chance nodes, we have provided a quadratic bound on the memory states needed to represent an equilbrium. However, we show that finding such an equilibrium is NP-hard. This motivates an approximation algorithm which for arbitrary ϵ computes a strategy which is at most ϵ alway from a Stackelberg equilibrium. The runtime of this algorithm depends cubically on the inverse of such ϵ.

By experimentally evaluating the approximation algorithm we tested it's scalability. We have found out that the bounds on memory are quite loose when considering randomly generated DAGs. It would be interesting to consider whether these bounds can be made tighter.

We left unadressed the problem of memory requirements for stochastic games with chance nodes. For pure strategies, it seems a reduction can be made from discounted sum games.

References

[1] Branislav Bošanskỳ, Simina Brânzei, Kristoffer Arnsfelt Hansen, Troels Bjerre Lund, and Peter Bro Miltersen. Computation of Stackelberg equilibria of finite sequential games. *ACM Transactions on Economics and Computation (TEAC)*. 2017, 5 (4), 23.

[2] Vincent Conitzer, and Tuomas Sandholm. *Computing the optimal strategy to commit to.* In: *Proceedings of the 7th ACM conference on Electronic commerce*. 2006. 82–90.

[3] Anshul Gupta. *Equilibria in Finite Games*. Ph.D. Thesis, University of Liverpool. 2015.

[4] Anshul Gupta, Sven Schewe, and Dominik Wojtczak. Making the best of limited memory in multi-player discounted sum games. *arXiv preprint arXiv:1410.4154*. 2014,

[5] John C Harsanyi. A new theory of equilibrium selection for games with complete information. *Games and Economic Behavior*. 1995, 8 (1), 91–122.

[6] H Kuhn. Extensive games and the problem op information. *InH. KuhnandA. Tucker, editors, Contributions to the Theory of Games*. 2016, 193–216.

[7] Joshua Letchford. *Computational aspects of stackelberg games*. Ph.D. Thesis, Duke University. 2013.

[8] Joshua Letchford, and Vincent Conitzer. *Computing optimal strategies to commit to in extensive-form games.* In: *Proceedings of the 11th ACM conference on Electronic commerce*. 2010. 83–92.

[9] Joshua Letchford, Liam MacDermed, Vincent Conitzer, Ronald Parr, and Charles L Isbell. *Computing Optimal Strategies to Commit to in Stochastic Games..* In: *AAAI*. 2012.

[10] John F Nash, and others. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*. 1950, 36 (1), 48–49.

[11] Martin J Osborne, and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[12] Yoav Shoham, and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[13] Milind Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[14] Heinrich Von Stackelberg. *Marktform und gleichgewicht*. J. springer, 1934.

[15] Bernhard Von Stengel, and Shmuel Zamir. *Leadership with commitment to mixed strategies*. . Technical Report LSE-CDAM-2004-01, CDAM Research Report.

Appendix A
Notation

- $\mathcal{S}, \mathcal{S}_1, \mathcal{S}_2, \mathcal{Z}$ the sets of inner nodes, leader's nodess, follower's nodes and leaves
- $\mathcal{A}, \mathcal{A}_1, \mathcal{A}_2$ the sets of all actions, actions of the leader and actions of the follower
- $\mathcal{A}(s)$ are the actions available in a given state
- $\pi = v_0 a_0 \dots v_{k-1} a_{k-1} v_k$ where v_0 is the root note is a history
- πa history derived from π by playing a after playing π
- $\pi[j]$ is the prefix of π up to v_j (but without a_j)
- $\pi[i{:}j]$ is the subsequence of π from v_i to v_j
- $\sigma(\pi)$ probability that π results when playing according to strategy profile σ
- $\sigma(\pi\downarrow)$ the strategy profile in the supgraph rooted by the last node of π, which follows after playing π
- $\mu(n)$ maxmin value of n
- M the set of memory states
- m_0 the initial memory state
- $\mathcal{M} : M \times (N \times A) \to M$ the memory update function
- $u_i^\sigma(\pi)$ utility of history π when playing according to σ
- $u_i^\sigma(n)$ utility of a node n if the strategy after n is independent of the history (the strategy is positional)
- $u_i^\sigma(m, n)$ utility of the node n reached with memory state m if the play continues according to σ.
- Π_i is the set of probability mixtures over strategies
- $\mathcal{BR}(\sigma_{-i})$ is the set of best responses of player i to strategy profile σ_{-i}
- $\square$ the proof is finished. Or also Elementary, my dear Watson.