UNIVERSITÄT DUISBURG-ESSEN
FAKULTÄT FÜR INGENIEURWISSENSCHAFTEN
ABTEILUNG INFORMATIK UND ANGEWANDTE KOGNITIONSWISSENSCHAFT

Master's Thesis

Infinite Games: Algorithms and Reductions

Maxime Nederkorn
Matrikelnummer: 3004376

UNIVERSITÄT
DUISBURG
ESSEN

Department of Computer Science and
Applied Cognitive Science
Faculty of Engineering
University of Duisburg-Essen

31. January 2022

Examiners:
Prof. Dr. Barbara König
Prof. Dr. Janis Voigtländer

Advisor:
Richard Eggert

Contents

1. Introduction

[EM79][ZP96][Con92][BFL+08][EJ91]There are practical applications (cite) which are able to be modeled by or reduced to various games. Considering the many ways there are to solve those games, the possibility to (repeatedly) reduce them to one another and various possibilities to implement any of those algorithms there are many potential means to solve problems on those games. We want to see how the different ways to solve problems, the reductions between those problems and the combination of the two play out and compare to one another in a real-world implementation.

1.1. Task

The games we concern ourselves here are *Parity Games* (PG), *Mean Payoff Games* (MPG), *Energy Games* (EG), *Discounted Payoff Games* (DPG) and (stopping) *Simple Stochastic Games* (SSG). For any of those games there are multiple problems, such as "what is the value of vertex v under optimal strategies from both players?" or "what are the optimal strategies?". For any of those problems, there are multiple ways to solve them for any specific game directly, some of which are abstractable to multiple of the games, such as Kleene Iteration and Strategy Iteration. Additionally we can reduce the games to another as following:

$$\begin{tikzcd}
PG \arrow[r] & MPG \arrow[r] \arrow[d] & DPG \arrow[r] & SSG \\
 & EG & &
\end{tikzcd}$$

The reductions themselves are computationally relatively simple compared to the problems. Some of them however produce graphs that are bigger in some way ($PG \to MPG$, $DPG \to SSG$), which increases the complexity of subsequently applied algorithms, or reduce to multiple other game problems. Another part we have control over is how exactly the algorithms are implemented. The biggest factor here being that, due to their nature, the edge-weights can be seen as an incidence matrix and can therefore make use of matrix operations as a form of parallelization for parts of the algorithms.

1.2. Structure

The paper is structured as follows:

In Chapter 2 we will first define the different kinds of games we are concerned with and the notion of *value* as on objective for the respective games as well as (memory-less) strategies as a means to achieve those values.
In Chapter 3 we will first explain in 3.1 the idea of value and strategy iteration and then both concrete implementations of those as well as other algorithms for the respective games. In 3.2 we will then show how the different games and their underlying graphs can be translated to one another and how the problems posed on those games are finally reduced.

In Chapter 4 we show the specifics as to how the theoretical algorithms were internally implemented and elaborate on the choices and decisions arising in the implementations of the reductions and solutions.
In Chapter 5 we show how the implementations embeds to be used to solve specific problems or to demonstrate with the GUI.
In Chaper 6 we show the kinds of graphs and problems we used to evaluate the solutions and the potential benefit of prior reduction as well as the results of the evaluation.
In Chapter 7 we interpret the results of the evaluation and give suggestions and ideas for further improvement.

2. Definitions

Foundational to our task are the different kinds of *Infinite Games* and how to determine the outcome of each such game. To do so, we also want a notion of *strategies* on such Infinite Games. To later reduce the games to one another, we furthermore want a definition of a *reduction* in the computational complexity sense.

2.1. Directed Graphs

Let V be a finite set of Vertices, let $E \subseteq V \times V$ be a set of Edges, then $G = (V, E)$ is a *Directed Graph.* We also define

$$\begin{aligned} \mathit{pre}\colon V &\to \mathcal{P}(V) & \mathit{pre}(v) &= \{u \in V \mid (u, v) \in E\} \\ \mathit{post}\colon V &\to \mathcal{P}(V) & \mathit{post}(v) &= \{u \in V \mid (v, u) \in E\} \end{aligned}$$

2.2. Arenas

An arena is an extension of Directed Graphs, where the set of Vertices, V, is partitioned into two disjunct subsets V_0 and V_1, respectively denoting the regions where player 0, also represented by $\square$, and player 1, also represented by $\bigcirc$, are to play. We also require that the out-degree of every vertex is at least one, so that any play on the Arena can always be prolonged.
Formally, let (V, E) be a non-trivial Directed Graph, $V_0 \cup V_1 = V,\ V_0 \cap V_1 = \emptyset$ be a partition of V and $\forall v \in V : \mathit{post}(v) \neq \emptyset$, then $A = (V, (V_0, V_1), E)$ is an Arena.

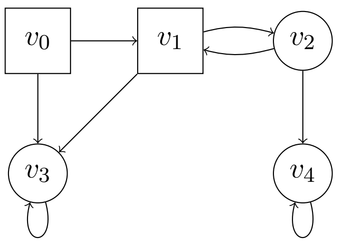

Figure 1: Arena A

Example 1:

An Arena

$$\begin{aligned} A = (&\{v_0, v_1, v_2, v_3, v_4\}, (\{v_0, v_1\}, \{v_2, v_3, v_4\}), \\ &\{(v_0, v_1), (v_0, v_3), (v_1, v_2), (v_2, v_1), \\ &(v_2, v_4), (v_4, v_4), (v_1, v_3), (v_3, v_3)\}) \end{aligned}$$

2.3. Positions, Moves

A *position*, $\pi_i = (v_0, v_1, \dots, v_i)$, in a Game describes a finite path on the underlying graph, i.e. a position is an element of V^+. E.g. in Fig. 1 starting at v_0, (v_0, v_1, v_2, v_4) could be a play.
A *move* is an extension of a position in the graph by one more step. E.g. in Fig. 1 $(v_0, v_1, v_2) \mapsto (v_0, v_1, v_2, v_4)$ could be a move. Player i chooses the n-th move $(\dots, v_{n-1}) \mapsto (\dots, v_{n-1}, v_n)$ for $v_{n-1} \in V_i \in (V_0, V_1)$.

2.4. Strategies

A *strategy*, $V^* \times V_i \to V$, is a function by which player i choses the next move for any given position.
We call a strategy *memoryless* if for any given position the next move only depends on the last vertex of the position, i.e. $V_i \to V$. We will refer to memoryless strategies of player 0 as σ and of player 1 as τ.
E.g. in Fig. 1 $\sigma = \{(v_0, v_1), (v_1, v_2)\}$ and $\tau = \{(v_2, v_1), (v_3, v_3), (v_4, v_4)\}$ could be strategies.

2.5. Plays

A *play* describes the path of arbitrary length $\langle v_0, v_1, \dots \rangle$ the player go trough in the process of playing the game. We refer to the play generated by applying strategies σ, τ to game G starting at $v_0 \in V$ as $\pi_{\sigma,\tau}(G, v_0) = \langle v_0, v_1, \dots \rangle$
E.g. if we take the previous example strategies and apply them to $\pi_{\sigma,\tau}(A, v_0)$ we get the play $\langle v_0, v_1, v_2, v_1, \dots \rangle$. The play can be arbitrarily prolonged by applying the moves $\sigma\colon v_1 \mapsto v_2$ and $\tau\colon v_2 \mapsto v_1$.

2.6. Infinite Games

Infinite Games are a category of games played by two players on a finite, directed graph. They are infinite in the sense that we require the out-degree of every vertex to be at least one. As such, regardless of the strategies chosen by the players, they never terminate.

2.6.1. Parity Games

Parity Games are played by two players, *Even* or player 0 ($\square$) and *Odd* or player 1 ($\bigcirc$). A Parity Game, $PG = (A, p)$, is played on an Arena A with a priority function $p\colon V \to \{0, 1, \dots, |V|\}$.
Let $\pi_{\sigma,\tau}(PG, v_0) = \langle v_0, v_1, \dots \rangle$ be the *play* resulting from applying the strategies σ and τ to Parity Game PG. Let

$$\#_\infty(\pi_{\sigma,\tau}(PG, v_0)) =$$
$$\{i \in \{0, 1, \dots, |V|\} \mid \forall j \in \mathbb{N}_0\colon \exists n \in \mathbb{N}_0\colon j < |\langle v \in \langle v_0, \dots, v_n \rangle\colon p(v) = i \rangle|\}$$

be the set of priorities that appear arbitrarily often in the play.
If $max(\#_\infty(\pi_{\sigma,\tau}(PG, v_0))) := v_{\sigma,\tau}(v_0)$, the value of vertex v_0 is even, then *Even* wins and vice versa. Optimal strategies for *Even*/*Odd* are those that result in the most/least starting vertices resulting in an even/odd value.

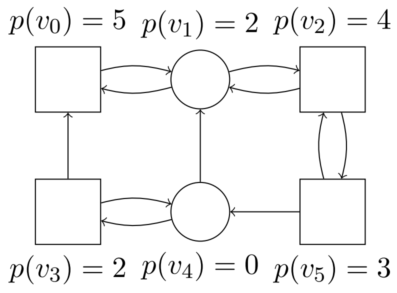

Figure 2: PG

Example 2:

Playing PG with $\sigma = \{(v_0, v_1), (v_2, v_5), (v_3, v_4), (v_5, v_2)\}$ and $\tau = \{(v_1, v_0), (v_4, v_1)\}$, the respective optimal strategies, results in play $\pi_{\sigma,\tau}(PG, v_0) = \langle v_0, v_1, v_0, ..\rangle$ for which $v_{\sigma,\tau}(v_0) = 5$ is odd and the play is therefore winning for *Odd*.

2.6.2. Mean Payoff Games

Mean Payoff Games are played by two players, *Max* or player 0 (□) and *Min* or player 1 (○). A Mean Payoff Game, $MPG = (A, \nu, d, w)$, is played on an Arena A, with threshold $\nu \in \mathbb{Z}$ and an edge-weight function $w\colon E \to \{-d, \dots, -1, 0, 1, \dots, d\}$, $d \in \mathbb{N}_0$.
Max wins play $\pi_{\sigma,\tau}(MPG, v_0) = \langle v_0, v_1, \dots\rangle$ if

$$\liminf_{n\to\infty} \left(\frac{1}{n} \sum_{i=0}^{n-1} w((v_i, v_{i+1})) \right) \geq \nu.$$

Alternatively we can also ask what, for a given play, the maximum threshold ν is with which *Max* can still win the play. We call this the *value* $v_{\sigma,\tau}(v)$ of a vertex v. The goal for *Max* is to maximize the mean payoff to guarantee a win under the highest possible ν and for *Min* to minimize the mean payoff as to guarantee a win under the lowest possible ν. Optimal strategies are those that maximize/minimize the mean payoff for a given MPG regardless of the initial vertex.

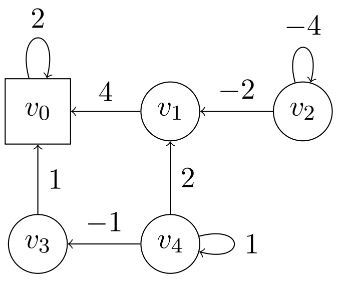

Figure 3: MPG

Example 3:

A Mean Payoff Game
$MPG = (A, 0, 4, \{((v_0, v_0), 2), ((v_1, v_0), 4), \dots\})$.
Playing MPG with $\sigma = \{(v_0, v_0)\}$ and
$\tau = \{(v_1, v_0), (v_2, v_2), (v_3, v_0), (v_4, v_4)\}$, the respective optimal strategies, results in the values $v_{\sigma,\tau} = \{(v_0, 2), (v_1, 2), (v_2, -4), (v_3, 2), (v_4, 1)\}$

2.6.3. Energy Games

Energy Games are played by two players, *Charging* or player 0 (□) and *Depleting* or player 1 (○). An Energy Game, $EG = (A, c, d, w)$, is played on an Arena A, with initial

credit $c \in \mathbb{N}_0$ and an edge-weight function $w\colon E \to \{-d, \dots, -1, 0, 1, \dots, d\}$, $d \in \mathbb{N}_0$.
Charging wins play $\pi_{\sigma,\tau}(EG, v_0) = \langle v_0, v_1, \dots\rangle$ if

$$\forall k \in \mathbb{N}_0\colon \left(\sum_{i=0}^{k} w((v_i, v_{i+1}))\right) + c \geq 0.$$

For any given position $\langle v_0, v_1, \dots, v_k\rangle$ in a play we call $\left(\sum_{i=0}^{k-1} w((v_i, v_{i+1}))\right) + c$ the *energy level* at that position. Keep in mind that *Charging* doesn't necessarily aim to maximise their energy level at *any* specific position but rather to maintain a non-negative energy level at *every* position. Alternatively we can also ask what, for a given play, the minimum initial credit c is with which *Charging* can maintain the winning condition. We call this the *minimum initial credit* or *value* $v_{\sigma,\tau}(v)$ of a vertex v. If no such value exists, e.g. when the vertex is of *Depleting* and has a self-loop with negative edge-weight, we write $v_{\sigma,\tau}(v) = \infty$. The goal for *Charging* is to avoid getting trapped in cycles with overall negative edge-weight, as these can deplete any initial credit given, as well as to minimize the initial credit necessary to the compliant with the winning condition in all other cycles. Conversely, the goal for *Depleting* is to trap *Charging* in negative cycles or at least to maximize the initial credit necessary for *Charging* to reach a non-negative cycle. Optimal strategies are those that minimize the initial credit necessary for *Charging* and those that either deny the winning condition entirely or maximize the initial credit necessary for *Depleting*.

Example 4:

Let us reappropriate Arena A and edge-weight function w of the previous MPG example and create an Energy Game $EG = (A, 0, 4, \{((v_0, v_0), 2), ((v_1, v_0), 4), \dots\})$.
Playing EG with $\sigma = \{(v_0, v_0)\}$ and $\tau = \{(v_1, v_0), (v_2, v_2), (v_3, v_0), (v_4, v_3)\}$, the respective optimal strategies, results in the values $v_{\sigma,\tau} = \{(v_0, 0), (v_1, 0), (v_2, \infty), (v_3, 0), (v_4, 1)\}$.

2.6.4. Discounted Payoff Games

Discounted Payoff Games are played by two players, *Max* or player 0 (□) and *Min* or player 1 (○). A Discounted Payoff Game, $DPG = (A, \nu, d, w, \lambda)$, is played on an Arena A, with threshold $\nu \in \mathbb{Z}$, an edge-weight function $w\colon E \to \{-d, \dots, -1, 0, 1, \dots, d\}$, $d \in \mathbb{N}_0$ and a discount factor $0 < \lambda < 1$.
Max wins play $\pi_{\sigma,\tau}(DPG, v_0) = \langle v_0, v_1, \dots\rangle$ if

$$(1 - \lambda)\left(\sum_{i=0}^{\infty} \lambda^i \cdot w((v_i, v_{i+1}))\right) \geq \nu.$$

Similar to MPGs we can also alternatively ask for a maximum winable threshold v with which *Max* can still win. Again we call this the *value* $v_{\sigma,\tau}(v)$ of a vertex v. Optimal strategies are also defined analogously to MPGs.

Example 5:

Let us again reappropriate Arena A and edge-weight function w and create a Discounted Payoff Game $DPG = (A, 0, 4, \{((v_0, v_0), 2), ((v_1, v_0), 4), \dots\}, 0.95)$.
Playing DPG with $\sigma = \{(v_0, v_0)\}$ and $\tau = \{(v_1, v_0), (v_2, v_2), (v_3, v_0), (v_4, v_4)\}$, the respective optimal strategies, results in the values $v_{\sigma,\tau} = \{(v_0, 2), (v_1, 2.1), (v_2, -4), (v_3, 1.95), (v_4, 1)\}$.

2.7. Simple Stochastic Games

Simple Stochastic Games are played by two players, *Max* or player 0 ($\square$) and *Min* or player 1 ($\bigcirc$). A Simple Stochastic Game, $SSG = (G, (V_0, V_1, V_2), \mathtt{0}, \mathtt{1}, p)$, is played on a Directed Graph G, with a partition (V_0, V_1, V_2), two sink vertices $\mathtt{0}, \mathtt{1}$ and a probability function $p\colon V_2 \to (V \to [0,1])$. We require that the probabilites of all outgoing edges of each V_2 vertex sum to 1: $\forall v \in V_2\colon \sum_{u \in V} p(v)(u) = 1$ and that $\forall (u,v) \in (V_2 \times V) \setminus E\colon p_u(v) = 0$. On vertices $v \in V_2$, also called *Average* or *Random* ($\Diamond$) vertices, the next vertex gets chosen according to the probability function $p_v\colon V \to [0,1]$ rather than a player. We also require, like for Arenas, that the out-degree of every vertex is at least one, with the exception of $\mathtt{0}, \mathtt{1}$, for which it is zero.
Max wins if the $\mathtt{1}$ sink is reached, *Min* wins if the $\mathtt{0}$ sink is reached or the game doesn't terminate. Since the result of a play $\pi_{\sigma,\tau}(SSG, v_0)$ can be probablistic, it is assigned a probabilty to reach the $\mathtt{1}$ sink rather than a fixed value. We say that a SSG is *stopping* if for every possible position in a play, so given two fixed strategies σ, τ, there is a path to one of the sink vertices. All SSGs we will consider from here on out will be stopping-SSGs. Since stopping-SSGs terminate, they are not Infinite Games. For simplicity, we will still include SSGs under the *Infinite Games* label from here on out. The goal for *Max* in a stopping-SSG is to maximize the chance to reach the $\mathtt{1}$ vertex, the goal for *Min* to reach the $\mathtt{0}$ vertex. Optimal strategies are those that maximize the chance to reach the desired sink-vertex of the respective player regardless of the initial vertex.

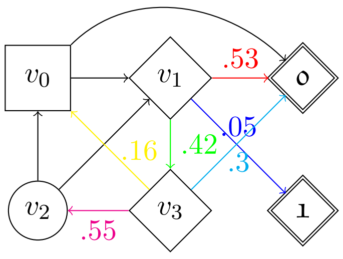

Figure 4: SSG

Example 6:

A Simple Stochastic Game

$$\begin{aligned} SSG = (G, (\{v_0\}, \{v_2\}, \{v_1, v_3\}), \mathtt{0}, \mathtt{1},\\ \{(v_1, \{(\mathtt{0}, .53), (\mathtt{1}, .05), (v_3, .42)\}),\\ (v_3, \{(\mathtt{0}, .3), (v_0, .16), (v_2, .55)\})\}). \end{aligned}$$

Playing SSG with $\sigma = \{(v_0, v_1)\}$ and $\tau = \{(v_2, v_0)\}$, the respective optimal strategies, results in the values $v_{\sigma,\tau} = \{(v_0, .07), (v_1, .07), (v_2, .07), (v_3, .05), (\mathtt{0}, 0), (\mathtt{1}, 1)\}$

2.8. Reductions

A *reduction* describes an algorithm with which we can transform (reduce) one class of problems to another. For our purposes, we choose target problem classes that are already solved. Aside from reducing to an already solved problem class and therefore, per definition, also solving the original problem class, reducing and subsequent solving of instances of the original problem may be faster than any attempt to directly solve the original problem without reducing to another intermediary infinite game problem.
Formally, we express our problem classes as formal languages A and B over the alphabets Σ^* and Γ^*. If f is totally computable and

$$f\colon \Sigma^* \to \Gamma^*, A \subseteq \Sigma^*, B \subseteq \Gamma^*$$
$$\forall w \in \Sigma^*\colon w \in A \iff f(w) \in B$$

then f is a reduction from Σ^* to Γ^*.
In practise our formal languages A and B will be some instances of types of infinite games together with statements about such games, such as the values under perfect play or perfect strategies. The reduction will then draw an equivalence to the other infinite game, e.g. if $w =$ "$v(v_u) = 5$ in G" is a word in A then $f(w) =$ "$v(v_v) = 16$ in H" is a word in B.

3. Solutions and Reductions in Theory

Before we go into the specifics of the implementation of the algorithms used to solve the problems and the reductions we first want to comprehensively explain the theory and reasoning behind each of them.

3.1. Solutions in Theory

One of the key parts of our work is the solving of problems on our games, be it in conjunction with a reduction or not. First we shall explain the general idea of Value Iteration and Strategy Iteration, as they can be generalized and used to solve multiple of the problems posed. Then we will explain the specific implementations of those as well as unique algorithms used to solve each of the games we are interested in.

3.1.1. Value Iteration

3.1.2. Strategy Iteration

3.1.3. PGs

3.1.4. MPGs

3.1.5. DPGs

3.1.6. EGs

3.1.7. SSGs

3.2. Reductions in Theory

3.2.1. PGs to MPGs

3.2.2. MPGs to DPGs

3.2.3. MPGs to EGs

3.2.4. DPGs to SSGs

4. Solutions and Reductions in Practise

4.1. Solutions in Practise

4.2. Reductions in Practise

5. Implementation

6. Evaluation

7. Conclusion and further approaches

A. Versicherung an Eides Statt

Ich versichere an Eides statt durch meine untenstehende Unterschrift,

- dass ich die vorliegende Arbeit - mit Ausnahme der Anleitung durch die Betreuer - selbstständig ohne fremde Hilfe angefertigt habe und
- dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus fremden Quellen entnommen sind, entsprechend als Zitate gekennzeichnet habe und
- dass ich ausschließlich die angegebenen Quellen (Literatur, Internetseiten, sonstige Hilfsmittel) verwendet habe und
- dass ich alle entsprechenden Angaben nach bestem Wissen und Gewissen vorgenommen habe, dass sie der Wahrheit entsprechen und dass ich nichts verschwiegen habe.

Mir ist bekannt, dass eine falsche Versicherung an Eides Statt nach § 156 und
nach § 163 Abs. 1 des Strafgesetzbuches mit Freiheitsstrafe oder Geldstrafe bestraft wird.

Ort, Datum	Unterschrift

References

[BFL$^+$08] Bouyer, Patricia ; Fahrenberg, Uli ; Larsen, Kim G. ; Markey, Nicolas ; Srba, Jiří: Infinite Runs in Weighted Timed Automata with Energy Constraints. In: Cassez, Franck (Hrsg.) ; Jard, Claude (Hrsg.): *Formal Modeling and Analysis of Timed Systems*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008. – ISBN 978–3–540–85778–5, S. 33–47

[Con92] Condon, Anne: The complexity of stochastic games. In: *Information and Computation* 96 (1992), Nr. 2, 203-224. `http://dx.doi.org/https://doi.org/10.1016/0890-5401(92)90048-K`. – DOI https://doi.org/10.1016/0890–5401(92)90048–K. – ISSN 0890–5401

[EJ91] Emerson, E.A. ; Jutla, C.S.: Tree automata, mu-calculus and determinacy. In: *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, 1991, S. 368–377

[EM79] Ehrenfeucht, A. ; Mycielski, J.: Positional strategies for mean payoff games. In: *International Journal of Game Theory* 8 (1979), Jun, Nr. 2, 109-113. `http://dx.doi.org/10.1007/BF01768705`. – DOI 10.1007/BF01768705. – ISSN 1432–1270

[ZP96] Zwick, Uri ; Paterson, Mike: The complexity of mean payoff games on graphs. In: *Theoretical Computer Science* 158 (1996), Nr. 1, 343-359. `http://dx.doi.org/https://doi.org/10.1016/0304-3975(95)00188-3`. – DOI https://doi.org/10.1016/0304–3975(95)00188–3. – ISSN 0304–3975