Strategy improvement method for solving simple stochastic games

Xavier Badin de Montjoye

université
PARIS-SACLAY

Strategy improvement method for solving simple stochastic games

Méthode d'amélioration de stratégie pour résoudre les jeux stochastiques simples

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique
Graduate School : Informatique et Science du numérique
Référent : Université de Versailles-Saint-Quentin-en-Yvelines

Thèse préparée dans l'unité de recherche DAVID (Université Paris-Saclay,UVSQ), sous la direction de Yann STROZECKI, maître de conférence, le co-encadrement de David AUGER, maître de conférence, et le co-encadrement de Jean-Michel FOURNEAU, Professeur

Thèse soutenue à Versailles, le 11 Juillet 2022, par

Xavier BADIN de MONTJOYE

Composition du jury

Nathalie BERTRAND Directrice de Recherche, INRIA Rennes - Bretagne Atlantique	Présidente
Anne CONDON Professeur, University of British Columbia	Rapporteur
Nathanaël FIJALKOW Chargé de recherche, CNRS, LaBRI, Université de Bordeaux	Rapporteur & Examinateur
Ana BUSIC Chargée de recherche, INRIA Paris	Examinatrice
Jean-Michel FOURNEAU Professeur, DAVID, UVSQ	Examinateur
Stéphane GAUBERT Directeur de recherche, INRIA Saclay	Examinateur
Yann STROZECKI Maître de conférence, DAVID, UVSQ	Directeur de thèse

THESE DE DOCTORAT
NNT : 2022UPASG047

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Méthodes d'amélioration de stratégies pour résoudre les jeux stochastiques simples

Mots clés : jeux stochastiques simples - théorie algorithmique des jeux - jeux stochastiques - Amélioration de stratégie

Résumé : Un Jeu Stochastique est joué par deux joueurs, MAX et MIN, en déplaçant un pion sur un ensemble d'états. À chaque tour, les deux joueurs choisissent simultanément une action et le pion est déplacé suivant une loi de probabilité dépendante du choix des deux joueurs. De plus, une valeur est associée à chaque état et le joueur MIN doit payer au joueur MAX cette valeur à chaque entrée dans l'état. Cela continue jusqu'à ce que le pion entre dans un puits. Dans ce cas, le jeu s'arrête. Il s'agit donc d'un jeu à somme nulle, où le but de MAX est de maximiser l'espérance de la valeur gagner durant la partie et MIN cherche à la minimiser. Ces jeux ont été introduits par Shapley en 1953, qui prouve qu'il existe des stratégies optimales pour les joueurs qui sont sans mémoire, et de ce fait, un gain moyen optimal.

En 1990, Condon présente une restriction des jeux stochastiques appelés jeux stochastiques simples (SSG pour Simple Stochastic Games en anglais) dans lequel les joueurs jouent à tour de rôle. Elle prouve qu'il existe une paire de stratégies déterministe, sans mémoire et optimales pour les joueurs. Notre but est de calculer ces stratégies. Pour le moment, les algorithmes déterministes pour résoudre ce problème sont en temps exponentiel, quant aux algorithmes non déterministes ils ont un temps moyen sous-exponentiel.

Dans cette thèse, on se focalise principale sur la méthode d'amélioration de stratégie pour résoudre des SSGs. On commence par reprouver certains résultats connus, mais sans utiliser la condition d'arrêt, une condition sur les SSG garantissant que la partie s'achève avec probabilité 1. Ensuite, on prouve une borne stricte sur le format des valeurs des sommets, et on présente un algorithme générique qui capture les algorithmes d'amélioration de stratégies. On l'utilise pour prouver bornes de complexités générales pour tous les algorithmes d'améliorations de stratégies. On compare également les méthodes d'améliorations de stratégies et d'itération de valeur. Dans un second temps, on introduit le concept de super-switch et on présente de nouveaux algorithmes récursifs pour résoudre les SSGs. On exprime ensuite le problème SSG comme un problème d'orientation à puits unique (USO) sur le permutahedron, un polytope dont les sommets sont les éléments du groupe symétrique. Enfin, on cherche à étendre une notion classique en théorie des jeux combinatoire dans le cadre des jeux stochastiques simples : la somme de jeux. On introduit un paramètre sur les jeux combinatoires, le plus court chemin ralenti, dans le but de sommer des versions randomisées de jeux impartiaux, et on le calcule pour quelques jeux classiques.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Strategy Improvement Methods for Solving Simple Stochastic Games
Keywords: algorithmic game theory - stochastic games - simple stochastic games - Strategy Improvement

Abstract: A Stochastic Game is played by two players, MAX and MIN, by moving a token on a set of states. At each step, both players simultaneously choose an action which moves the token to a new state according to a probability distribution dependent on both actions. In addition, player MIN has to pay to player MAX a value depending on the new state. This continues until the token ends in a sink, in which case the game stops. Hence, this is a zero-sum game where MAX seeks to maximise its expected value gained during the game and MIN tries to minimise it. Those games were introduced by Shapley in 1953 who proved that there exists a pair of memoryless optimal strategies for the players and thus an optimal expected gain.

In 1990, Condon presented a restriction of Stochastic Games called Simple Stochastic Games (SSG) in which players play alternatively. They proved that there exist a pair of optimal strategies that are memoryless and determinist. Our goal is to compute those strategies. At the moment, all deterministic algorithms for solving this problem are exponential and non-deterministic algorithms are in sub-exponential time.

In this thesis, we mainly focus on the strategy improvement method for solving SSGs. We first reprove some known results without using the stopping condition, a condition on SSG guaranteeing that the game stops with probability 1. Then, we prove a bound on the format value of the vertices, and we present a generic algorithm to describe strategy improvement algorithms. We use it in order to prove some time bounds for all instances of strategy improvement algorithms. Then we compare the strategy improvement and the value iteration method. After that, we introduce the concept of super-switch and provide new recursive algorithms for solving SSG. We then express the SSG problem as a unique sink orientation problem on the permutahedron, a polytope whose set of vertices is the symmetric group. Finally, we try to extend the classic notion of summing games in combinatorial game theory to extend it in the case of simple stochastic games. We introduce a parameter on combinatorial games, the slowed shortest path, in order to sum randomised version of impartial games, and we compute it for some classical games.

Remerciements

Il me faut commencer par remercier Yann Strozecki, qui a eu la patience de m'encadrer d'abord en stage puis en thèse. Il m'a poussé à approfondir nombre de mes idées souvent déconstruites et m'a laissé une liberté d'éparpillement dont j'avais besoin. Et pour cela, je lui en suis reconnaissant. Merci à mes co-encadrants, David Auger pour ses discussions devant le tableau blanc, et Jean-Michel Fourneau pour son aide au cours de ses trois années. Merci à Pierre Coucheney pour toutes ces discussions sur les SSGs.

Je remercie chaleureusement Anne Condon et Nathanaël Fijalkow qui ont accepté d'être rapporteurs et m'ont permis d'améliorer ce mémoire de thèse. Enfin, je remercie les membres du jury qui ont pris le temps de venir m'écouter.

Contents

List of Figures

Introduction

One of the key questions behind the development of **game theory** is: how to gain the most advantage against someone whose goal is the total opposite of ours? More specifically, game theory tries to understand how two agents or players, with different goals, interact. The most classical formalisation of this situation is two-player zero-sum games, situations with two agents where the result is an advantage for one and an equivalent disadvantage for the other. For instance, in chess, the result is either a win for a player and a loss for the other, or a draw for both sides. At the casino, all the money earned by the player is a loss for the establishment and a loss of the player is a gain for the establishment. The mathematical aspect behind game theory is generally attributed to Zermelo (1913) [Zer13], Borel (1921) [Bor21] and Morgenstern and von Neumann (1944) [VNM07]. However, the study of games can be traced to much older works. It is generally admitted that probability theory stems from Cardano, in its book *Liber de ludo aleae* which translates to *Book on Games of Chance* and from the correspondence between Pascal and Fermat while trying to answer the question asked by Gombauld (Chevalier de Méré) on how to divide the bet in a game of luck that was prematurely stopped and when one of the player has a clear advantage against their opponent [Que53]. It is quite amusing to note that considering prematurely stopped game is an important tool that we use at several occasions in this thesis.

The use of game theory is not restricted to how to win board games, but has widespread impact on numerous fields. The most classic one is economy and more specifically microeconomics with 11 Nobel Prizes in Economics Science for game theorists, the last in date being for Wilson and Milgrom for their works in auction theory, a branch of game theory. Game theory also has uses in biology, with the field of evolutionary game theory, which comes from Maynard Smith and Price in 1973 [SP73]. Another application of game theory is political science, for instance to explain peace between democratic countries [LR04].

Due to the vastness of game theory, most fields focus on specific versions of games. The first possible choice is the number of players. Another point is the amount of information each player has access to. If all the agents have access to the same information, then it is a perfect information game. Otherwise, it is not. For instance, in poker each player has information that the others do not have, namely the content of their hands. In a game of Rock Paper Scissors, although both players do not know what the other will play, they still have the same amount of information when choosing their action. It leads us to another distinction: simultaneous and sequential games. In simultaneous games, both players make an action at the same time, which gives a result depending on both actions. In sequential games, players play alternatively, as it is the case in Monopoly, Backgammon or tic-tac-toe. Finally, it is important to distinguish between probabilistic games and deterministic games, that is between games where randomness appears, as in black-jack, and games where it does not, as in the game of go or rock paper scissors.

A **Stochastic Game** is played by two players, MAX and MIN, by moving a token on a set of states. At each step, both players simultaneously choose an action which moves the token to a new state according to a probability distribution dependent on both actions. In addition, player MIN has to pay to player MAX a value depending on the new state. This continues until the token ends in a sink, in which case the game stops. Hence, this is a zero-sum game where MAX seeks to maximise its expected value gained during the game and MIN tries to minimise it. An instance of stochastic games is two players betting with each other on a die roll. Another instance is someone playing at the casino. The states of the game being defined as

the amount of money of the player and the casino and the actions being for the player to either continue or quit and for the casino, to let them continue or to ban them. At each turn, the value of the sink corresponds to the gain of the bet. Stochastic Games were introduced by Shapley in 1953 [Sha53], where they proved that there exists a pair of memoryless optimal strategies for the players and thus an optimal expected gain. This defines for each state a value corresponding to the expected gain for player MAX when the game starts in this state. They propose a way to compute those optimal values using a method called value iteration or value propagation. The idea behind this method is to associate to each state a value and then propagate this value to the other states. The computation of this value answers the question of the Chevalier de Méré, and represents which value should be paid if the game is cut short in this state.

In 1990, Condon presented a restriction of Stochastic Games called Simple Stochastic Games (SSG) [Con92] in which players play alternatively. The probability distributions are represented by random nodes and the other nodes are divided between MAX and MIN nodes, which defines which player moves the token. Moreover, there are two sinks which are the only vertices with an associated value, the values being 0 and 1. Hence, Simple Stochastic Games are reachability games where MAX tries to maximise its probability to reach the sink of highest value. Classical games that can be modelled using Simple Stochastic Games are backgammon, the Ludo game, or Monopoly. In all these classical board games, the possible states of the games are finite, due to the finite nature of the material and the limited ways to place it. Moreover, each player does an action sequentially, and the random vertices represent the dice rolls. Condon proved that there exists a pair of optimal strategies for SSGs that are memoryless and deterministic. However, this result has been proven using the stopping condition. This condition requires that for each possible strategy, the game ends with probability 1. In Chapter 1 of this thesis, we lift the stopping hypothesis and reprove classical results for SSGs. We also introduce the concept of null set that needs to be studied when the game is not stopping. Moreover, in Chapter 3, we provide a way to remove the stopping condition in algorithms that were proven true only for stopping SSGs [TVK11]. These results are part of a paper [ABdMS21] presented at MFCS.

One of the main motivation for introducing Simple Stochastic Games was to study their computation power. In [CKS81], Chandra, Kozen and Stockmeyer present the model of **alternating Turing machine**. In this model, there are two different types of states, existential and universal. A word is accepted by an alternating Turing machine if, for all possible transitions from universal states, there is a path to an accepting state. In [Pap85], Papadimitriou presents another model called Game Against Nature, in which there are two different types of states, existential and random. In this model, when the machine reaches a random node, the next state is chosen following some probability distribution. A word is thus accepted if there is a path that reaches an accepting state with probability greater than 0.5. Games against nature extend the concept of probabilistic automaton introduced by Rabin in 1963 [Rab63]. In [Con92], Condon has introduced Simple Stochastic Games to study a mix of both models: alternating Turing machines with random, universal and existential nodes.

Simple stochastic games have multiple real life applications, such as autonomous **urban driving**. In [CKSW13], Chen, Kwiatkowska, Simaitis and Wiltsche modelled the problem of driving a car by a simple stochastic game where player MAX is the driver, MIN is the environment, and the random states represent the possibilities of hazards occurring. In each road segment, the environment selects possible hazards, and the driver has to choose a set of reactions to avoid them, while still trying to reach the destination. Another use is smart energy management [CFK+13]. Simple stochastic games can also be used for model checking

of the modal $\mu-$calculus as shown in [Sti99]. The idea behind it is to consider a new game called the modal checking game and then show the reduction to SSG.

The decision problem of deciding whether, starting from some vertex x, there is a strategy for MAX that makes the token reach the sink 1 with probability greater than 0.5 is in NP $\cap$ co-NP. No polynomial algorithm is known to solve this problem. The associated functional problem of finding an optimal pair of strategies is not known to be in FP, but is known to be in PPAD [Jub05]. The class PPAD, for Polynomial Parity Arguments on Directed graphs, was introduced by Papadimitriou in [Pap94]. This class of problems corresponds to the set of problems that can be polynomially reduced to the End-of-the-Line problem: given a boolean circuit that generates a graph such that each vertex has at most one predecessor and one successor, and given a vertex without any predecessor, the goal is to find a vertex with no predecessor or no successor. To be more precise, the functional SSG problem is included in USO which will be defined later, which itself is in UEOPL, the class of problems complete for the Unique End of Potential Line problem, a modification of the End of the Line problem with an added valuation on vertices [FGMS20].

If SSG can be considered to be at the bottom of the PPAD class, some celebrated families of two player games can be reduced to SSGs. For instance, **Parity Games** that were introduced under that name by Emerson and Jutla in [EJ91]. This game has no random vertex and no sink, and both player try to visit infinitely often different vertices. To be more precise, an integer is assigned to each vertex as a priority. The goal of player one is that the vertex visited infinitely often with the highest priority is odd, and the goal of player two is that its priority is even. A quasi-polynomial algorithm has been found by Calude, Jain, Khoussainov, Li and Stephan in 2017 for solving parity games [CJK+20]. This family can be reduced to two other families of games: **Mean Payoff Games** and **Discounted Games**, which can be viewed as sequential stochastic games with no randomness. Those games can be solved in pseudo-polynomial time, as Zwick and Paterson showed in [ZP96]. Those games can be reduced to SSG [Pur97, ZP96, CF11]. We briefly present this in Chapter 1. Moreover, Anderson and Miltersen showed [AM09] that SSG is polynomial time equivalent to randomized variants of these classic games.

As stated before, the first method to solve Simple Stochastic Games is the **value iteration algorithm** that originates from Stochastic Games [Sha53]. In the case of SSG with r random nodes with probability distribution $(1/2, 1/2)$, Ibsen-Jensen and Miltersen presented in 2012 a value iteration running in time $O^*(2^r)$ where O^* does not count polynomial factors. We study this family of algorithm in Chapter 2 and to give a good bound on their complexity, we present a tight bound on the representation of the value of an SSG.

The second method is the **strategy improvement method** that came from Howard in [How60] who introduced it in order to solve Markov Decision Process. **Markov Decision Process** were introduced by Bellman in 1957 [Bel57] and as it was later realised, it can be interpreted as a stochastic game with no MIN player. In 1966, Hoffmann and Karp presented a strategy improvement algorithm in order to solve Markov Decision Process [HK66]. This was further studied in the case of SSG by Tripathi, Valkanova and Kumar in 2011 [TVK11]. Friedmann offered in 2009 [Fri11] an exponential lower bound of this algorithm using parity games and Fearnley adapted it directly to Markov decision process [Fea10]. Some strategy improvement algorithms are based on the notion of switch, which consists in locally improving a strategy in order to obtain a globally better strategy. In Chapter 3, we provide a general framework called GSIA to study this kind of algorithm. We also present a new tool called the max of two strategies, such that given two strategies, one can find a strategy that is better than both strategies. We mostly focus on deterministic algorithms that

are all in exponential time. Ludwig found a non-deterministic strategy improvement algorithm in 1995 for solving Simple Stochastic Games in sub-exponential expected time [Lud95].

In Chapter 4, we present some classical algorithm in the context of GSIA. We study the Gimbert and Horn algorithm (GHA) [GH08] which relies on finding the optimal order on the random vertices, and we compare it to the Ibsen-Jensen and Miltersen's algorithm (IJMA). We show that GHA needs strictly fewer iterations than IJMA, although the iterations of IJMA are less costly. We then present an improvment of IJMA that needs strictly less runtime.

Then, in Chapter 5, we present a family of **recursive** algorithms that we introduced in [BdM21]. The idea behind these algorithms is the new concept of super-switch, which consists in fixing the strategy in some vertices and optimally solving the rest of the game. We present two algorithms, one that partitions the set of vertices into pairs of vertices and the other one that fixes a decreasing set of vertices. Both algorithms have an exponential complexity, which is better than the complexity of strategy improvement or value iteration algorithms presented above. Those algorithms can be seen as variation of algorithm on USO, that we define below. However, they have a worse complexity bound than algorithms designed for solving USO problems.

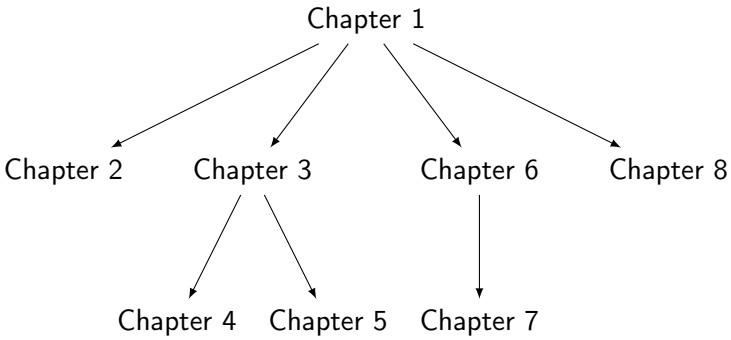
The functional problem of SSG can be reduced to a **Unique Sink Orientation** (USO) problem on cubes [BSV03, BSV04, GJMR08]. The USO problem introduced by Stickney and Watson [SW78] is the following one. Given a polytope and an orientation of its edges such that each face of the polytope has exactly one sink vertex: a vertex with no outgoing edges, find the sink of the polytope. The idea to prove that SSG can be seen as a USO problem is to associate to each possible strategy for player MAX a vertex of a polytope. There is one optimal strategy which corresponds to the sink, and each face of our polytope models a variation of our original game with some edges being removed. Szabó and Welzl present in [SW01] an algorithm for solving the USO problem on cubes called the Fibonacci Seesaw Algorithm. This algorithm has a complexity in $O^*(1.61^n)$ with n the dimension of the cubes which is for SSG the number of vertices associated with player MAX. This is the best bound in n known for solving SSGs with a deterministic algorithm. Barba, Milatz, Nummenpalo, Sun, Thomas, Zhang and Zhang [BMN+19] have presented an algorithm for solving USO on grids. We present these algorithms as well as a reduction from SSG to USO in Chapter 6. Then, we prove in Chapter 7 that SSG can be interpreted as a USO problem on the **permutahedron**, a polytope whose set of vertices is the symmetric group. In this case, we do not define each vertex as a strategy, but as the value vector of a modification of our games that forces an order for random vertices. Those subgames are such that for one of them, its optimal value vector corresponds to the optimal value vector of our original game.

Finally, in Chapter 8, we study the possibility to sum SSGs. Summing games is a classic notion in **combinatorial game theory** [Con00]. Combinatorial game theory studies deterministic games with perfect information, where players play alternatively in order to reach some position. Those game include for instance chess, checkers or the game of go. In some games, for instance in go, the board can often be divided in smaller independent games. Since the player can only make one move, they have to make it in one of the subgames. In go, these small subgames carry a concept of initiative with positions having *sente* or *gote* a notion that is somehow similar to sum of games. A sum of games can be seen as placing multiple games next to each others and being able to play in only one of them at each time, or, more formally, it can also be viewed as studying the Cartesian product of games. A category of combinatorial games are impartial games: games where the possible moves only depend on the position and not whose turn it is. The

canonical example of such games are subtraction games, in which a pile of tokens is placed between two players and each player removes some tokens until the pile is empty. Those games can be easily summed using the Sprague-Grundy Theorem. Our goal is to extend such results to stochastic version of these games. We introduced a variant of SSG that we call Impartial Simple Stochastic Games (ISSG) in order to sum games and then study a restricted version of ISSG. An important aspect to study sums of games, is the possibility to give a small description of a game that is exponentially larger. For instance in chess, the short description corresponds to the set of rules, but the graph of the games is the graph of every possible board positions, which is not realistic to study. We prove that understanding this restricted class requires finding the slowed shortest path in a combinatorial game, a tool that we introduced in Chapter 8. This corresponds to the path where the winning player wants to end the fastest and the losing player wants to make the game last as long as possible. We show that the value of the sum of two ISSGs cannot be easily expressed.

In conclusion, in this thesis, we reprove some classic results and correctness of algorithms without using the restriction of the stopping condition in Chapter 1, 2 and 3. Moreover, we provide a tight bound on the value format of the nodes in Chapter 2. We provide in Chapter 3 a Generic Algorithm to describe strategy improvement algorithms and obtain generic time bounds for all strategy improvement algorithms. This Generic Algorithm allows us to easily consider new instances of strategy improvement. In Chapter 4, we compare strategy improvement and value iteration algorithm. Then, we provide in Chapter 5 two new strategy improvement algorithms that are recursive and work on non-stopping SSG. Then, we prove a reduction in Chapter 7 of the functional SSG problem to a USO problem on the permutahedron by considering vertices representing subgames and not strategies. Finally, we look at the possibilities to study SSG with a vertex set exponential in their description by trying to extend the concept of summing combinatorial games to SSGs in Chapter 8.

We give a visual representation of how the chapters interact with each other:



Introduction (en Français)

Une des questions principales derrière le développement de la **théorie des jeux** est : comment obtenir le plus gros avantage possible contre quelqu'un dont le but est diamétralement opposé au nôtre ? Plus précisément, la théorie des jeux cherche à comprendre comment deux agents, appelés aussi joueurs, aux buts différents interagissent. La formalisation la plus classique de cette situation sont les jeux à deux joueurs à somme nulle. Ce sont des situations avec deux agents dont le résultat final est un avantage pour l'un et un désavantage de valeur équivalente pour l'autre. Par exemple, aux échecs, le résultat d'une partie est soit une victoire pour l'un et une défaite pour l'autre, soit une égalité. Il n'est pas possible d'avoir une victoire des deux joueurs. Au casino, tout l'argent gagné par un joueur est une perte pour l'établissement. L'aspect mathématique derrière la théorie des jeux est généralement attribué à Zermelo (1913) [Zer13], Borel (1921) [Bor21] et Morgenstern et von Neumann (1944) [VNM07]. Cependant, l'étude des jeux se retrouve dans des travaux bien antérieurs. Par exemple, il est généralement admis que la théorie des probabilités vient de Cardano, dans un livre intitulé en latin *Liber de ludo aleae* qui se traduit par *Livre sur les jeux de chances*. Une autre origine des probabilités tient des correspondances entre Pascal et Fermat qui cherchaient à répondre à une question posée par Gombauld, le Chevalier de Méré, sur comment diviser l'enjeu d'un jeu de hasard arrêté de façon prématurée lorsque l'un des joueurs a un clair avantage sur son adversaire. Il est amusant de noter que le principe de jeu arrêté prématurément est un outil important que l'on retrouve à plusieurs occasions tout au long de cette thèse.

L'utilité de la théorie des jeux n'est pas restreinte à la question de comment gagner à des jeux de sociétés, mais possède un impact important sur de nombreux domaines. Le plus connu étant sans doute l'économie et plus précisément la microéconomie avec 11 prix Nobel d'économie revenant à des travaux sur la théorie des jeux. Le dernier en date étant pour Wilson et Milgrom pour leurs travaux sur la théorie des enchères, une branche de la théorie des jeux. La théorie des jeux trouve également des utilités en biologie avec le domaine de la théorie évolutive des jeux provenant de Maynard Smith et Price en 1973 [SP73]. Une autre application de la théorie des jeux se retrouve dans les sciences politiques, par exemple pour expliquer la paix entre deux démocraties [LR04].

Du fait de l'étendue de la théorie des jeux, la majorité des domaines s'y rapportant se concentre sur des versions spécifiques de jeux. La première considération possible est le nombre de joueurs. Un autre point est la quantité d'informations dont chaque joueur a accès. Si tous les agents ont accès à la même information, on dit que le jeu est à information parfaite. Par exemple, lors d'une partie de poker, chaque joueur a accès à une information que les autres n'ont pas : le contenu de leurs mains. Lors d'une partie de Pierre Feuille Ciseaux, bien que chaque joueur ignore ce que son adversaire va jouer, ils ont tout de même accès à la même information lors du choix d'action. Cela nous mène à une autre distinction : les jeux simultanés et les jeux séquentiels. Dans le cas des jeux simultanés, chaque joueur effectue une action au même moment, ce qui donne un résultat dépendant de toutes les actions. Dans les jeux séquentiels, les joueurs jouent à tour de rôle, comme c'est le cas pour le Monopoly, le Backgammon ou encore le morpion. Enfin, il est important de différencier les jeux probabilistes et déterministes. C'est-à-dire, entre les jeux où le hasard apparaît, comme c'est le cas au black jack, et où il n'apparaît pas, comme c'est le cas à pierre-papier-ciseaux.

Un **Jeu Stochastique** est joué par deux joueurs, MAX et MIN, en déplaçant un jeton sur un ensemble d'états. À chaque étape, les joueurs choisissent simultanément une action et le jeton est déplacé suivant

une distribution de probabilité dépendante de ces deux actions. De plus, le joueur MIN doit payer au joueur MAX une somme dépendant du nouvel état atteint. Cela continue jusqu'à ce que le jeton atteigne un puits, ce qui cause la fin du jeu. Ainsi, c'est un jeu à somme nul où MAX cherche à maximiser son espérance de gains durant la partie et MIN cherche à la minimiser. Un exemple de jeu stochastique est deux joueurs pariant sur un lancer de dé. Un autre exemple est quelqu'un jouant au casino. Les états du jeu sont définis comme les sommes possibles de fond du joueur et du casino et les actions possibles étant pour le joueur soit de continuer, soit de s'arrêter, pour le casino, soit de le laisser jouer, soit de le bannir. À chaque tour, la valeur de l'état correspond aux gains du dernier pari. Les jeux stochastiques ont été introduits par Shapley en 1953 [Sha53], ou il prouve l'existence d'une paire de stratégies optimales sans mémoire pour les deux joueurs et donc une espérance de gain optimale. Cela définit pour chaque état une valeur correspondant à l'espérance de gain du joueur MAX lorsque le jeu débute dans cet état. Shapley propose une façon de calculer ces valeurs en utilisant une méthode appelée itération de valeur ou propagation de valeur. L'idée derrière cette méthode est d'associer à chaque état une valeur et ensuite propager cette valeur aux états adjacents. On peut noter que ces valeurs répondent à la question du Chevalier de Méré et représentent quelle valeur devrait être payée si le jeu était arrêté avant la fin dans cet état.

En 1990, Condon présente une restriction des Jeux Stochastiques appelée **Jeux Stochastiques Simples** [Con92] (noté SSG pour *Simple Stochastic Games*) dans laquelle les joueurs agissent séquentiellement. Les distributions de probabilités sont représentées par des sommets aléatoires et les autres sommets sont divisés en sommet MAX et MIN. Ils définissent lequel des deux joueurs joue. De plus, il y a deux puits qui sont les seuls sommets avec une valeur associée, celles-ci étant 0 et 1. Ainsi, les Jeux Stochastiques Simples sont des jeux d'accessibilités où MAX essayent de maximiser la probabilité d'atteindre le puits de valeur 1. Comme jeux classiques pouvant être modélisés par un jeu stochastique simple, on peut noter l'exemple du backgammon, des petits-chevaux ou encore du Monopoly à deux joueurs. Dans tous ces jeux de plateaux classiques, le nombre d'états possibles est fini du fait du caractère fini du matériel et des options limités d'où le matériel peut-être placé. De plus, chaque joueur agit de façon séquentielle et les sommets aléatoires représentent les lancers de dés. Condon prouve qu'il existe une paire de stratégies optimale pour cette famille de jeu qui sont déterministes et sans mémoire. Cependant, ce résultat a été prouvé en utilisant la condition d'arrêt. Cette condition impose que quelque soit les stratégies choisies par chaque joueur, le jeu s'arrête, c'est-à-dire le jeton atteint un puits, avec probabilité 1. Dans le Chapitre 1 de cette thèse, on supprime l'hypothèse d'arrêt et on reprouve plusieurs résultats classiques sur les SSGs sans l'utiliser. On introduit également le concept d'ensemble nul qui doit être pris en compte dans le cas des jeux ne satisfaisant pas la condition d'arrêt. De plus, au Chapitre 3, on donne une façon de retirer la condition d'arrêt dans des algorithmes qui ont été prouvés corrects uniquement pour les SSGs satisfaisant la condition d'arrêt [TVK11]. Ces résultats font partis d'un papier [ABdMS21] présenté à MFCS.

Une des motivations principales pour introduire les jeux stochastiques simples est l'étude de leur puissance de calcul. Dans [CKS81], Chandra, Kozen et Stockmeyer présentent le modèle de **machine de Turing alternante**. Dans ce modèle, il y a deux types d'états, existentiels et universels. Un mot est accepté par une machine de Turing alternante si pour toutes les transitions possibles à partir d'états universels, il existe un chemin jusqu'à un état acceptant. Dans [Pap85], Papadimitriou présente un autre modèle appelé Jeu Contre la Nature, dans lequel une machine de Turing a deux types d'états, existentiels et aléatoires. Dans ce modèle, lorsque la machine atteint un état aléatoire, le prochain état est choisi suivant une certaine loi de probabilité. Ainsi, un mot est accepté s'il existe un chemin qui atteint un état acceptant avec probabilité

strictement plus grande que 0.5. Les jeux contre la nature étendent le concept d'automates probabilistes introduits par Rabin en 1963 [Rab63]. Dans [Con92], Condon a introduit les jeux stochastiques simples pour étudier un mix de ces deux modèles, des machines de Turing alternantes avec des états aléatoires, universels et existentiels.

Les Jeux stochastiques simples ont plusieurs utilités pratiques comme la **conduite urbaine** de véhicules autonomes. Dans [CKSW13], Chen, Kwiatkowska, Simaitis et Wiltsche modélisent le problème de conduire une voiture par un jeu stochastique simple où le joueur MAX est le conducteur, le joueur MIN représente l'environnement et les états aléatoires représentent la probabilité d'apparitions de dangers. Pour chaque segment de route, l'environnement sélectionne une liste de dangers possibles et le conducteur doit effectuer une succession de réactions pour les éviter tout en tentant d'atteindre sa destination. Une autre utilité des jeux stochastiques simples consiste à la gestion intelligente d'énergie [CFK+13]. Les jeux stochastiques simples peuvent aussi être utilisés pour la vérification de modèle pour la logique du $\mu-$calcul modal comme montré dans [Sti99]. L'idée derrière étant de considérer un nouveau jeu appelé le jeu de vérification modale et d'ensuite effectuer une réduction vers SSG.

Le problème décisionnel de savoir si, en commençant dans un état x, il existe une stratégie MAX pour laquelle le jeton atteint un sommet 1 avec probabilité strictement plus grande que 0.5 est dans NP $\cap$ co-NP. Aucun algorithme polynomial est connu pour le moment pour résoudre ce problème. Le problème fonctionnel associé consistant à trouver une paire de stratégies optimales n'a pas été prouvé comme appartenant à FP, mais appartient à la classe PPAD [Jub05]. La classe PPAD, pour Polynomial Parity Arguments on Directed graphs, a été introduite par Papadimitriou dans [Pap94]. Cette classe de problèmes correspond à l'ensemble des problèmes qui peuvent être réduits polynomialement au problème de la fin de ligne. Étant donné un circuit booléen qui génère un graphe tel que chaque sommet a au plus un prédécesseur et un successeur, et étant donné un sommet sans prédécesseur, le but est de trouver un autre sommet sans prédécesseur ou sans successeur. Pour être plus précis, le problème fonctionnel associé aux SSGs est inclus dans USO, que nous définirons plus tard, qui est elle-même inclus dans UEOPL, la classe des problèmes complets pour le problème de fin de ligne à potentielle unique, une modification du problème de fin de ligne avec une valuation supplémentaire sur les sommets [FGMS20].

Si les SSG peuvent être considérés comme étant en bas de la classe PPAD, quelques familles célèbres de SSGs peuvent être réduites à SSG. Par exemple, les **Jeux de Parités** qui ont été introduits sous ce nom par Emerson et Jutla dans [EJ91]. Ces jeux n'ont pas de sommets aléatoires, ni de puits, et chaque joueur tente de visiter infiniment souvent différents ensembles de sommets. Pour être plus précis, un entier est assigné comme priorité à chaque sommet. Le but du joueur 1 est que le sommet visité infiniment souvent avec la plus haute priorité ait une priorité impaire et le joueur 2 cherche à ce qu'elle soit paire. Un algorithme quasi polynomial a été trouvé par Calude, Jain, Khoussainov, Li et Stephan en 2017 pour résoudre les jeux de parités [CJK+20]. Cette famille peut être réduite vers deux autres familles de jeux : **les jeux à gains moyens** et **les jeux à gain amortis**, qui peuvent être considérés comme des jeux stochastiques séquentiels et sans hasard. Ces jeux peuvent être résolus en tant pseudo-polynomial comme l'ont montré Zwick et Paterson dans [ZP96]. Ces jeux peuvent être réduits à SSG [Pur97, ZP96, CF11]. On présente brièvement ces réductions dans le Chapitre 1. De plus, Anderson et Miltersen ont prouvé [AM09] que les SSG sont polynomialement équivalent aux versions randomisées de ces jeux.

Comme précisé plus haut, la première méthode pour résoudre des jeux stochastiques simples est **l'algorithme d'itération de valeurs** provenant des jeux stochastiques [Sha53]. Dans le cas des SSG avec r sommets

aléatoires et des distributions de probabilités $(1/2, 1/2)$, Ibsen-Jensen et Miltersen ont présentés en 2012 un algorithme d'itération de valeur en temps $O^*\left(2^r\right)$ où O^* ne comptent pas les facteurs polynomiaux. On étudie cette famille d'algorithme au Chapitre 2 et on donne une borne sur le format des valeurs d'un SSG.

La deuxième méthode est la méthode **d'amélioration de stratégie** qui provient de Howard dans [How60] qui l'introduit pour résoudre les processus de décision Markovien. Les **processus de décision Markovien** ont été introduits par Bellman en 1957 [Bel57], et, comme cela fut remarqué plus tard, peuvent être interprétés comme des jeux stochastiques sans joueur MIN. En 1966, Hoffmann et Karp présentent un algorithme d'amélioration de stratégie pour résoudre les processus de décision Markovien [HK66]. Cet algorithme a été étudié plus en détail dans le cas des SSG par Tripathi, Valkanova et Kumar en 2011 [TVK11]. Friedmann présente en 2009 [Fri11] une borne inférieure exponentielle pour cet algorithme en utilisant des jeux de parités et Fearnley adapte cette preuve directement dans le cas des processus de décision Markovien [Fea10]. Certains algorithmes d'amélioration de stratégies sont basés sur la notion de switch, qui consiste à améliorer localement une stratégie pour obtenir une stratégie globalement meilleure. Au Chapitre 3, on présente un cadre général, que l'on appelle GSIA, pour étudier cette famille d'algorithme. On présente également un nouvel outil appelé max de deux stratégies qui, étant donné deux stratégies, nous donne une stratégie meilleure que ces deux stratégies. On s'intéresse principalement aux algorithmes déterministes qui tournent tous en temps exponentiel. Ludwig présente en 1995 un algorithme d'amélioration de stratégie non déterministe qui tourne en temps moyen sous-exponentiel.

Au Chapitre 4, on présente plusieurs algorithmes classiques dans le contexte de GSIA. On étudie l'algorithme de Gimbert et Horn (GHA) [GH08] qui consiste à ordonner les sommets aléatoires, et on le compare avec l'algorithme d'itération de valeur d'Ibsen-Jensen et Miltersen (IJMA). On montre que GHA demande strictement moins d'itérations que IJMA, bien que les itérations de IJMA soient moins coûteuses. On présente alors une amélioration de IJMA.

Ensuite, au Chapitre 5, on présente une famille d'algorithmes **récursifs** que nous avons introduite dans [BdM21]. L'idée derrière ces algorithmes est le nouveau concept de super-switch, qui consiste à fixer la stratégie sur certains sommets et d'ensuite résoudre optimalement le reste du jeu. On présente deux algorithmes, un qui partitionne l'ensemble des sommets en pairs de sommets et l'autre qui fixe un ensemble décroissant de sommets. Ces deux algorithmes ont une complexité exponentielle qui est meilleure que les algorithmes présentés plus haut. Ces algorithmes peuvent être regardés comme une variation d'algorithmes sur les USO, problème que nous définissons plus bas. Cependant, ils ont une pire complexité que les algorithmes construits pour résoudre les problèmes sur les USOs.

Le problème fonctionnel sur les SSG peut être réduit au problème d'orientation de puits unique (USO pour Unique Sink Orientation) sur les cubes [BSV03, BSV04, GJMR08]. Le problème d'orientation de puits unique introduit par Stickney et Watson [SW78] est le suivant. Étant donné un polytope et une orientation de ses arêtes telle que chaque face du polytope a exactement un seul puits, un sommet sans arête sortante, trouver le puits du polytope global. L'idée derrière la réduction de SSG vers USO est d'associer à chaque stratégie possible du joueur MAX un sommet du polytope. Il existe une meilleure stratégie qui correspond à un puits et chaque face de notre polytope modélise une modification de notre jeu originale où certaines arêtes ont été supprimées. Szabó and Welzl présente un algorithme pour résoudre les problèmes USO sur les cubes appelé l'algorithme à bascule de Fibonacci. Cet algorithme a une complexité en $O^*\left(1.61^n\right)$ avec n la dimension du cube, ce qui correspond dans le cas des SSG aux nombres de sommets MAX. Il s'agit de la meilleure borne dépendant de n connu pour résoudre les SSGs de degré 2 de façon déterministe.

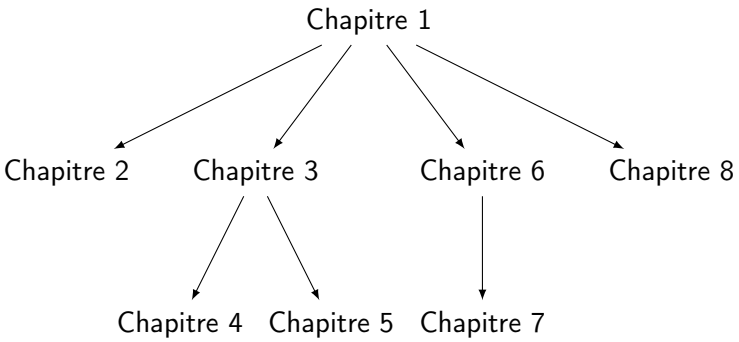
Barba, Milatz, Nummenpalo, Sun, Thomas, Zhang et Zhang [BMN+19] ont présenté un algorithme pour résoudre des USO sur les grilles. On présente ces algorithmes ainsi qu'une réduction de SSG vers USO au Chapitre 6. Ensuite, on prouve au Chapitre 7 que SSG peut être interprété comme un problème USO sur le **permutahedron**, un polytope dont l'ensemble de sommets corresponds aux éléments du groupe symétriques : aux permutations. Dans ce cas, on ne définit pas chaque sommet comme une stratégie, mais comme un vecteur de valeur d'une modification du jeu forçant un ordre sur les sommets aléatoires. Ces sous-jeux sont tels que pour l'un d'entre eux, son vecteur de valeur optimal correspond au vecteur de valeur optimale du jeu original.

Enfin, au Chapitre 8, on étudie la possibilité de sommer des SSGs. Sommer des jeux est une notion classique en **théorie des jeux combinatoire** [Con00]. La théorie des jeux combinatoires étudie des jeux à deux joueurs déterministes à informations complètes, où les joueurs jouent alternativement pour atteindre une certaine position. Ces jeux incluent par exemple les échecs, les dames ou le jeu de go. Dans certains jeux, par exemple au go, le plateau de jeu peut être divisé en plusieurs sous-jeux indépendants. Puisque les joueurs peuvent faire qu'un seul coup, ils doivent choisir dans quel sous-jeu ils veulent agir. Au go, ces sous-jeux contiennent un principe d'initiative avec certaines positions pouvant être soit *sente* ou *gote*, une notion qui se rapproche de celui de somme de jeu. Une somme de jeux peut être vue comme placer plusieurs jeux les uns à côté des autres et n'autoriser de jouer que dans un d'entre eux, ou, plus formellement, il s'agit d'étudier le produit Cartésien de jeux. Une sous-catégorie des jeux combinatoire sont les jeux impartiaux : des jeux pour lesquels les déplacements possibles ne dépendent que de la position et non pas de quel joueur doit jouer. L'exemple canonique de cette famille de jeux est les jeux de soustractions. Une pile d'allumette est placée entre deux joueurs et chaque joueur enlève un certain nombre d'allumettes suivant une liste de règles jusqu'à ce que la pile soit vide. Ces jeux peuvent être facilement sommés en utilisant le théorème de Sprague-Grundy. Notre but est d'étendre ces résultats à une version stochastiques de ces jeux. On introduit donc une variante des SSG appelés jeux stochastiques simples impartiaux (ISSG) et on étudie ensuite une restriction des ISSGs. Un point important dans l'étude des sommes de jeux est la possibilité de considérer une courte description d'un jeu exponentiellement plus grand. Par exemple, aux échecs, une courte description correspond à la liste des règles, mais le graphe du jeu est un graphe dont les sommets correspondent à chaque position possible du plateau, ce qui ne peut être réalistiquement étudié. On prouve que comprendre cette classe restreinte de ISSG nécessite d'étudier le plus court chemin ralentit pour les jeux combinatoires, un outil qu'on introduit au Chapitre 8. Cela correspond au chemin lorsque le joueur en position gagnante veut terminer le plus vite possible et son adversaire veut faire durer la partie le plus longtemps possible. On montre que la valeur d'une somme de deux ISSGs ne peut pas être facilement exprimée.

En conclusion, dans cette thèse, on reprouve quelques résultats classiques ainsi que la correction d'algorithmes connus sans utiliser la condition d'arrêt dans les Chapitres 1, 2 et 3. De plus, on donne une borne serrée sur le format des valeurs des sommets d'un jeu au Chapitre 2. On présente au Chapitre 3 un algorithme générique pour décrire les algorithmes d'amélioration de stratégies et obtenir des bornes en temps générique pour toute cette famille d'algorithmes. Cet algorithme générique nous permet d'aisément considérer de nouvelles instances d'amélioration de stratégies. Au Chapitre 4, on compare les algorithmes d'amélioration de stratégies et d'itération de valeurs. Ensuite, on présente au Chapitre 5 deux nouveaux algorithmes d'amélioration de stratégies qui sont récursifs et fonctionne sur des SSG ne satisfaisant pas la condition d'arrêt. Ensuite, on prouve une réduction au Chapitre 7 du problème fonctionnel sur les SSG

vers un problème USO sur le permutahedron en considérant les sommets comme des sous-jeux et non des stratégies. Enfin, au Chapitre 8, on s'intéresse à la possibilité d'étudier des SSGs avec un ensemble de sommets exponentiel en sa description en essayant d'étendre le concept de somme de jeux de la théorie des jeux combinatoires aux SSGs.

On donne une représentation de comment les chapitres s'articulent les uns avec les autres :



1 - Simple Stochastic Games and Strategies

Contents

1.1 . History and Definition

1.1.1 . History

In 1953, Shapley introduced the notion of stochastic games in [Sha53]. A Stochastic Game (SG) is played by two players MAX and MIN in a set of states. A token is positioned in some state and a value is associated to each state. Then, both players choose simultaneously an action and the token is randomly moved according to some probability distribution parametrised by both actions. When entering a state, player MIN pays the value associated to the state to MAX. The goal for MAX is thus to maximise their expected gain, while the goal of MIN is to minimise it. They proved that there exists a pair of optimal strategies and thus an optimal expected gain that can be computed by value iteration.

In 1957 [Bel57], Bellman introduced the concept of Markov Decision Process (MDP) that can either be interpreted as a Markov Chain with an added player, or as a stochastic game with one player. Howard then introduced the concept of policy iteration [How60] which consists in choosing some strategy for the player then improving on it until finding the best one. In this thesis, we also refer to policy iteration algorithms as strategy improvement algorithms.

Condon presents in 1990 a restriction of Stochastic Games in [Con92] called Simple Stochastic Games (SSG). In this version, players play alternatively, thus their optimal strategy is memoryless and deterministic. This was already claimed in the original paper of Shapley without any proof. In SSG, each state has different roles, either it is a random node that moves the token according to some probability distribution, or it belongs to one of the players, who decides where to move the token, or it is a sink in which the game stops. Moreover, the sinks are the only vertices with an associated value. In addition, non-sink vertices have out-degree 2 and all probability distributions are $(1/2, 1/2)$. In [Con90], the three principal families of algorithms for solving Simple Stochastic Games are presented: value iteration, strategy improvement and quadratic programming. They also present some natural but incorrect algorithms, while providing a proof of non-correctness. One of the most important examples being that the algorithm that considers the best response to the best response does not converge. In 2005, Somla presented an algorithm that does not directly fit in those families by using a geometric interpretation of the hypercube representing the different strategies of player MAX [Som05].

One of the main motivations for introducing Simple Stochastic Games was to study the computation power of the complexity model of space bound alternating Turing machine with random, universal and existential nodes. Informally, this corresponds to Turing Machines with $\exists$, $\forall$ and random states and a word is accepted if there exists a path in $\exists$ states, such that for all transitions in $\forall$ states there is probability greater than $1/2$ to end in an accepting state. This is a generalisation of the alternating Turing machine model introduced in [CKS81] which does not have random nodes and Game Against Nature [Pap85] which does not have universal nodes.

Under Condon's definitions, Ludwig provides in 1995 the first randomised sub-exponential algorithm to solve SSG [Lud95]. For SSG of higher degree Halman proves in 2007 that they can also be solved by a randomised algorithm in sub exponential time [Hal07] by considering SSG as an LP-Type problem and using the subexponential algorithm for LP-Type problem of Matoušek, Sharir and Welzl [MSW96]. For deterministic algorithms and n the number of vertices belonging to player MAX, Tripathi, Valkanova and Kumar [TVK11] show in 2011 a bound in $O\left(2^n/n\right)$ iterations for the Hoffman-Karp strategy improvement algorithm originally invented for solving Markov Decision Process [HK66].

For SSG with r a small number of random vertices but with arbitrary degree for all vertices, Gimbert and Horn gave in 2008 a strategy improvement algorithm needing $O(r!)$ iterations [GH08] by considering all possible ordering for the random vertices. Dai and Ge gave in 2009 ([DG09]) a randomised version of their algorithm by simply randomly picking an initial strategy, giving an expected running time of $O(\sqrt{r}!)$ iterations. In 2019, Auger, Coucheney and Strozecki gave a randomised algorithm in $2^{O(r)}$ [ACS19]. For SSG with random vertices of degree 2 and probability $(1/2, 1/2)$, Ibsen-Jensen and Miltersen [IJM12] gave a deterministic value iteration algorithm running in time $O^*(2^r)$ where O^* does not count polynomial factors.

Auger, Coucheney and Strozecki also studied in [ACS14] restricted classes of SSG, showing that solving SSG for a family of games called MAX-acyclic can be done in polynomial time. Moreover, they provide an FPT-algorithm parametrised by the number of fork vertices. Finally, they give a polynomial algorithm for SSG with a fixed number of feedback vertices using dichotomy method.

Lower bounds are known for some of those algorithms. For the value iteration algorithm of Ibsen-Jensen and Miltersen [IJM12], the time complexity is tight, as proved in their paper. In [Fri11], Friedmann gave an exponential lower bound for the Hoffman-Karp algorithm using parity games and Fearnley adapted it to Markov Decision Process [Fea10]. It is interesting to notice that, to the best of our knowledge, no lower bound is proved in the context of SSG but always in the simpler case of Markov Decision Process and Parity Games. However, we know that Markov Decision Process can be solved in polynomial time and Parity Game can be solved in quasi-polynomial time [CJK+20].

The problem of finding an optimal strategy of an SSG can be modelled by an acyclic unique sink orientation problem (USO) on cubes for SSG of degree 2 and on grids for higher degree [BSV03, BSV04, GJMR08]. The unique sink orientation problem is, given an orientation of the edges of a polytope such that each face has exactly one sink, find the global sink. The transformation considered is to regard the different strategies as vertices of a polytope. Notice that in [BSV03, BSV04], Björklund, Sandberg and Vorobyov do not use the term unique sink orientation but use the similar concept of Completely Local-Global functions. Szabó and Welzl present in [SW01] the Fibonacci Seesaw algorithm: a deterministic algorithm that solves USO on cubes of dimension n, and thus SSG with n MAX vertices, in $O^*(1.61^n)$ which is the best current bound for solving SSG. In [BdM21] we independently presented a version of this algorithm restricted to SSG. For SSG of higher degree d, Barba, Milatz, Nummenpalo, Sun, Thomas, Zhang and Zhang [BMN+19] present an algorithm visiting $O\left(d^{\lceil n/2 \rceil}\right)$ vertices. It is important to highlight that the reduction from SSG to USO is well known in the USO community, but not in the field of SSG, proof being that none of the papers previously cited about SSG referred to USO.

1.1.2 . Definition

We give a definition of SSG slightly more general than Condon in [Con90, Con92]. It is the same that we used in [ABdMS21].

Definition 1.1 (SSG). *A Simple Stochastic Game (SSG) is a directed graph G, together with:*

1. *A partition of the vertex set V in four parts V_{MAX}, V_{MIN}, V_R and V_S (all possibly empty, except V_S), satisfying the following conditions:*
   - *(a) every vertex of V_{MAX}, V_{MIN} or V_R has at least one outgoing arc;*
   - *(b) every vertex of V_S has exactly one outgoing arc which is a loop on itself.*

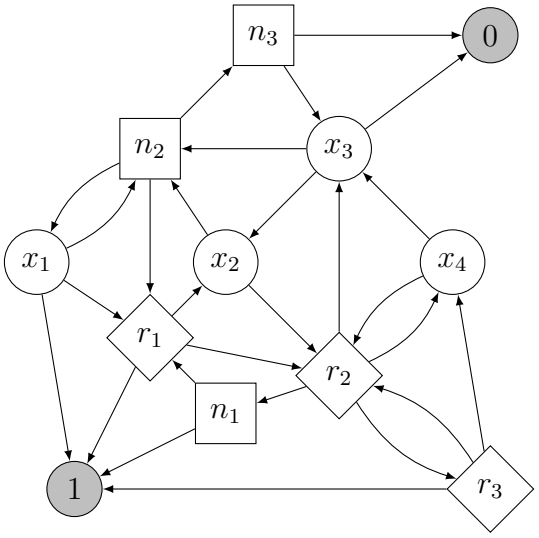


Figure 1.1: Instance of an SSG where square, circle and diamond vertices are respectively MAX, MIN and random vertices, and grey vertices are sink vertices. The probability distribution of each random vertex is the uniform distribution over their children.

2. *For every $x \in V_R$, a probability distribution $p_x(\cdot)$ with rational values, on the out-neighbourhood of x.*

3. *For every $x \in V_S$, a value Val(x) which is a rational number in the closed interval $[0, 1]$. We say that the value of x is Val(x) and that x is a Val(x)-sink.*

An example of an SSG is provided in Figure 1.1. The loops on the sinks are not represented.
We now introduce a few notations that will be useful in the rest of the thesis.

- We denote $|V_{\text{MAX}}|$ by n and $|V_R|$ by r.
- Vertices from V_{MAX}, V_{MIN}, V_R and V_S are respectively called MAX vertices, MIN vertices, random vertices and sinks.
- For $x \in V$, we denote by $N^+(x)$ the set of out-neighbours of x.
- We assume that for every $x \in V_R$ and $y \in V$, $y \in N^+(x)$ if and only if $p_x(y) > 0$.

It is useful to consider several families of SSG parametrised by the degree of their vertices or the value of their probability distribution. First, we present a restricted class of SSG defined by the degree of its MAX vertices.

Definition 1.2 (Binary SSG)**.** *A binary SSG is an SSG where every vertex of V_{MAX} has out-degree 2. An SSG is of degree d if its* MAX *vertices are of degree at most d.*

Now, we present a restricted class of SSG parametrised by the format of its probability distribution.

Definition 1.3 (q-SSG). *For q a positive integer, we say that an SSG is a q-SSG if there are only two sinks of value 0 and 1, and for all $x \in V_R$, there is an integer $q_x \leq q$ such that the probability distribution $p_x(\cdot)$ can be written as $p_x(x') = \frac{\ell_{x,x'}}{q_x}$ for all x' where $\ell_{x,x'}$ is a natural number.*

With these notations, the original definition of SSG by Condon in [Con92] corresponds to binary 2-SSG.

1.1.3 . How the Game is Played

First, we give an informal definition of how the game is played. The two players are named MAX and MIN. A token is positioned on a starting vertex x. If x is in V_{MAX} (resp. V_{MIN}) the MAX player (resp. the MIN player) chooses one of the out-neighbours of x to move the token to. If x is in V_R, the token is randomly moved to one of the out-neighbours of x according to the probability distribution $p_x(\cdot)$, independently of everything else. This process continues until the token reaches a sink s and then, player MIN has to pay $\mathsf{Val}(s)$ to player MAX and the game stops. Thus, the goal of player MIN is to minimise this value, while the goal of player MAX is to maximise it. The problem we study is to *find the best possible strategies* for MIN and MAX and the expected value that MIN has to pay to MAX while following those strategies.

Play and History

Definition 1.4 (Play). *A* **play** *in G is an infinite word $X = x_0x_1x_2\ldots$ on V^ω such that for all $t \geq 0$, (x_t, x_{t+1}) is an edge of G.*

If for a play $X = x_0x_1x_2\ldots$ there is some $t \geq 0$ with $x_t = s \in V_S$, then all subsequent vertices in the play are also equal to s. In this case, we say that the play **reaches** sink vertex s and we define the **value of the play** $\mathsf{Val}(X)$ as $\mathsf{Val}(s)$. If the play reaches no sink, then we set $\mathsf{Val}(X) = 0$.

Definition 1.5 (History). *A* **history** *of G is a finite word $h = x_0x_1\ldots x_k$ in V^*. If the last vertex x_k is a* MAX *vertex (resp.* MIN *vertex), we say that h is a* MAX *history (resp.* MIN *history).*

General Strategies

We now present the different kinds of strategies used to model how the actions of each player.

Definition 1.6 (General Strategy). *A* **general** MAX **strategy** *(resp. general* MIN *strategy) is a map σ assigning to every* MAX *history (resp.* MIN *history) $h = (x_0, x_1, \cdots, x_k)$ a probability distribution $\sigma(h)(\cdot)$ on the out-neighbourhood of x_k. The set of these strategies is denoted by $\Sigma_{gen}^{\text{MAX}}$ (resp. $\Sigma_{gen}^{\text{MIN}}$).*

Thus, a general MAX strategy represents how the player MAX will play, by taking into account everything that happened in the game so far.

For $\sigma \in \Sigma_{gen}^{\text{MAX}}$ and $\tau \in \Sigma_{gen}^{\text{MIN}}$, given a starting vertex x_0, we recursively define a random play $X = (X_0, X_1, \cdots)$ of G in the following way. At $t = 0$ let $X_0 = x_0$, and for $t \geq 0$:

- if $X_t \in V_{\text{MAX}}$, X_{t+1} is an out-neighbour of X_t chosen following the probability distribution $\sigma(h)(\cdot)$, independently of everything else;
- if $X_t \in V_{\text{MIN}}$, X_{t+1} is an out-neighbour of X_t chosen following the probability distribution $\tau(h)(\cdot)$, independently of everything else;

- if $X_t \in V_R$, then X_{t+1} is an out-neighbour of X_t chosen following the probability distribution $p_{X_t}(\cdot)$, independently of everything else;
- if $X_t \in V_S$, define $X_{t+1} = X_t$.

This defines a distribution on plays, which we denote by $\mathbb{P}^{x_0}_{\sigma,\tau}(\cdot)$, or simply $\mathbb{P}(\cdot)$ if strategies and starting vertex are clear from context. The corresponding expected value and conditional expected values are denoted by $\mathbb{E}^{x_0}_{\sigma,\tau}(\cdot|\cdot)$, or simply $\mathbb{E}(\cdot|\cdot)$.

Restricted Classes of Strategies

There are two different properties on general strategies that can be imposed. First, these strategies have memory and depend on the history. Second, these strategies are randomised. We consider memoryless strategies: strategies that do not depend on the history. Implicitly, we consider strategies to be memoryless.

Definition 1.7 (Memoryless Strategy)**.** *A* **memoryless** MAX **strategy** *(resp. memoryless* MIN *strategy) is a map σ assigning to every* MAX *vertex (resp.* MIN *vertices) x a probability distribution $\sigma(x)(\cdot)$ on the out-neighbourhood of x. The set of these strategies is denoted by Σ^{MAX}_{ML} (resp. Σ^{MIN}_{ML}).*

It is known that for Stochastic Games, there exist memoryless randomised optimal strategies [Sha53].

Definition 1.8 (Pure strategies)**.** *A* **pure** MAX **strategy** *(resp. pure* MIN *strategy) is a map σ assigning to every* MAX *history (resp.* MIN *history) $h = (x_0, x_1, \cdots, x_k)$ a vertex $\sigma(h)$ which is an out-neighbour of x_k. The set of these strategies is denoted by $\Sigma^{\text{MAX}}_{pure}$ (resp. $\Sigma^{\text{MIN}}_{pure}$).*

Pure strategies will be used only as an intermediate object in proofs but not as solutions of SSGs, for instance in Section 3.2.1. Lastly, we consider positional strategies which depend only on the last vertex of the history.

Definition 1.9 (Positional Strategies)**.** *A* **positional** MAX **strategy** *(resp. positional* MIN *strategy) is a map σ assigning to every* MAX *vertex (resp.* MIN *vertices) x a vertex $\sigma(x)$ which is an out-neighbour of x. The set of these strategies is denoted by Σ^{MAX} (resp. Σ^{MIN}).*

Positional strategies are pure memoryless strategies. We will see that it is enough to only consider positional strategies. Thus, if nothing is specified, we will always assume that a MAX strategy refers to a positional MAX strategy. By abusing the notation, it is possible to identify positional strategies as memoryless, pure or general strategies. In that case, we have the following inclusions:

$$\begin{array}{ccc} & \Sigma^{\text{MAX}}_{pure} & \\ \subseteq\!\!\!/ & & \subseteq \\ \Sigma^{\text{MAX}} & & \Sigma^{\text{MAX}}_{gen} \\ \subseteq & & \subseteq\!\!\!/ \\ & \Sigma^{\text{MAX}}_{ML} & \end{array}$$

Values in an SSG

Definition 1.10 (Value Vector)**.** *Let G be an SSG and let (σ, τ) be a pair of* MAX *and* MIN *strategies,* **the value vector** $v^G_{\sigma,\tau}$ *is the real vector of dimension $|V|$ defined by, for any $x_0 \in V$,*

$$v^G_{\sigma,\tau}(x_0) = \mathbb{E}^{x_0}_{\sigma,\tau}\left(Val(X)\right).$$

This value represents the **expected gains** *for player* MAX *if both players play according to (σ, τ) and the game starts in vertex x_0.*

The superscript G can be omitted when it is clear from the context.

To compare value vectors, we use the pointwise order: we say that $v \geq v'$ if for all vertices $x \in V$, we have $v(x) \geq v'(x)$. Moreover, we say that $v > v'$ if $v \geq v'$ and there is some x such that $v(x) > v'(x)$.

Definition 1.11 (Best Response)**.** *Given a* MAX *strategy σ, a* **best response** *to σ is a* MIN *strategy τ such that $v_{\sigma,\tau} \leq v_{\sigma,\tau'}$ for all* MIN *strategies τ'. We write $BR(\sigma)$ the set of positional best responses to σ.*

Similarly, a **best response** to a MIN strategy τ is a MAX strategy σ such that $v_{\sigma,\tau} \geq v_{\sigma',\tau}$ for all MAX strategies σ'.

Definition 1.12 (Optimal Strategy)**.** *Let G be an SSG, a* MAX *strategy σ^* is said to be optimal if for all $x \in V$:*

$$\inf_{\tau \in \Sigma^{\text{MIN}}_{gen}} v_{\sigma^*,\tau}(x) = \sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} \inf_{\tau \in \Sigma^{\text{MIN}}_{gen}} v_{\sigma,\tau}(x)$$

Similarly, a MIN *strategy τ^* is said to be optimal if for all $x \in V$:*

$$\sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} v_{\sigma,\tau^*}(x) = \inf_{\tau \in \Sigma^{\text{MIN}}_{gen}} \sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} v_{\sigma,\tau}(x)$$

In the rest of this chapter, we will show that there exist a pair of optimal strategies (σ^*, τ^*) that are positional and that satisfy for all x in V:

$$v_{\sigma^*,\tau^*}(x) = \sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} \inf_{\tau \in \Sigma^{\text{MIN}}_{gen}} v_{\sigma,\tau}(x) = \inf_{\tau \in \Sigma^{\text{MIN}}_{gen}} \sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} v_{\sigma,\tau}(x)$$

1.2 . Expressing the Problem

The decision problem DSSG that we are trying to solve is the following one: for G an SSG and x a vertex of G, is there a MAX strategy σ such that for any MIN strategy τ, $v_{\sigma,\tau}(x) > 0.5$? This problem is in NP ∩ co-NP and no polynomial algorithm is known to solve it.

We will see in the later part of this chapter that there is a pair of optimal strategies (σ^*, τ^*). The functional problem FSSG is the following one: for G an SSG, find (σ^*, τ^*) an optimal pair of strategies of G. FSSG is not known to be in FP, but it is in PPAD [Jub05].

In [Con92], Condon showed that DSSG for SSGs with no MAX vertices, and thus discrete reachability Markov Decision Process, is complete for the class of GAN-SPACE($\log(n)$) which corresponds to the complexity class of log space game against nature. Game against nature, that was introduced by Papadimitriou

in [Pap85] represents informally Turing machines with additional random states. The problem of DSSG for general SSGs is complete for the complexity class of log space randomised alternating Turing machines: an extension of the alternating Turing machine of [CKS81] where we consider Turing machines with existential, universal and randomised nodes.

1.3 . Properties

1.3.1 . The Stopping Condition

Simple stochastic games are often referred to as reachability games. In this definition, there is no 0-sink and just a single 1-sink that is the objective. Thus, player MIN has to enter a loop in order to win. We can immediately notice that in the same way, with our definition, 0 sinks are not necessary since encountering a loop also gives value 0. Therefore, it is important to detect the set of vertices in a game G that have value 0.

Definition 1.13 (Null Set)**.** *The* **null set** *of a game G is K^G, the set of vertices x such that there exist a* MIN *strategy τ such that for all* MAX *strategies σ, $v_{\sigma,\tau}(x) = 0$. For a* MAX *strategy σ of G and a* MIN *strategy τ, we call null set under (σ, τ), the set of vertices $K^G_{\sigma,\tau}$ with value zero under the pair of strategies (σ, τ). We call null set under σ the set K^G_σ of vertices x such that there exist a* MIN *strategy τ that satisfy $v_{\sigma,\tau}(x) = 0$.*

K^G can be computed in polynomial time with a **graph traversal algorithm** presented in Algorithm 1. If σ and τ are memoryless strategies, $K^G_{\sigma,\tau}$ and K^G_σ can similarly be computed. Once again, we omit the superscript G when the context is obvious. A MAX strategy is said to be positive if for every MIN strategy τ and every $x \notin K^G$, $v_{\sigma,\tau}(x) > 0$.

Conversely, it can also be useful, especially in order to simplify proofs, to consider games that end in a sink with probability 1 independently of the strategies of both players. This is called **the stopping property**. In Chapter 2, we show that every SSG can be approximated by an SSG satisfying the stopping property. Although this condition simplifies a lot of proofs, in this thesis, we prove that most results do not require the stopping condition to hold.

1.3.2 . The Value of Positional and Memoryless Strategies

For any pair of memoryless strategies (σ, τ), the value vector $v_{\sigma,\tau}$ can be computed in polynomial time. Indeed, by definition of fixing a strategy, we can replace the playable vertices with a random vertex with associated probability distribution the probability distribution of σ and τ. Hence, computing $v_{\sigma,\tau}$ is the same as solving a Markov chain, which can be done in polynomial time. Moreover, the fact that a vector is the value vector associated with $v_{\sigma,\tau}$ can be verified in linear time in the number of edges.

Lemma 1.14. *Let G be an SSG, (σ, τ) a pair of memoryless strategies and let v a vector of dimension $|V|$ that satisfies:*

1. *For any $x \in K_{\sigma,\tau}$, $v(x) = 0$*
2. *For any $x \in V_{\text{MAX}}$, $v(x) = \sum_{y \in N^+(x)} \sigma(x)(y)v(y)$*

Algorithm 1: Computing K

```
Data: G = (V, E) a SSG
Result: K the null set of G, σ a positive MAX strategy
begin
    M ⟵ {s ∈ V_S | Val(s) > 0}
    while x ∉ M and (x, y) ∈ E with y ∈ M do
        if x ∈ V_MAX then
            M ⟵ M ∪ {x}
            σ(x) = y
        if x ∈ V_R then
            M ⟵ M ∪ {x}
        if x ∈ V_MIN then
            if |N⁺(x)| = 1 then
                M ⟵ M ∪ {x}
            else
                Remove the edge (x, y)
    return V ∖ M,σ
```

3. *For any* $x \in V_{\text{MIN}}$, $v(x) = \sum_{y \in N^+(x)} \tau(x)(y)v(y)$

4. *For any* $x \in V_R$, $v(x) = \sum_{y \in N^+(x)} p_x(y)v(y)$

5. *For any* $x \in V_S$, $v(x) = Val(x)$

then $v = v_{\sigma,\tau}$.

We recall that for a memoryless strategy σ, $\sigma(x)(y)$ corresponds to the probability to go to y from x according to the strategy σ.

Proof. Let σ and τ be two memoryless strategies. Starting from vertices $x \in V_{\text{MAX}}$, the expected gain for player MAX $v_{\sigma,\tau}$ is $\sum_{y \in N^+(x)} \sigma(x)(y)v_{\sigma,\tau}(y)$ using the law of total probability. Similarly, for $x \in V_{\text{MIN}}$, $v_{\sigma,\tau}(x) = \sum_{y \in N^+(x)} \tau(x)(y)v_{\sigma,\tau}(y)$ and for $x \in V_R$, $v_{\sigma,\tau}(x) = \sum_{y \in N^+(x)} p_x(y)v_{\sigma,\tau}(y)$. For every vertex $x \in K_{\sigma,\tau}$, we replace the vertex with a sink of value 0. This does not change the value of G under σ, τ. We can then consider that under strategies (σ, τ), every playable vertex of G has a non-zero probability to reach a sink. For $m = |V \setminus V_S|$ if we consider the following square matrix Q of dimension $m \times m$

defined by:

$$Q_{i,j} = \begin{cases} \sigma(i)(j) \text{ if } i \in V_{\text{MAX}} \text{ and } j \in N^+(i) \\ \tau(i)(j) \text{ if } i \in V_{\text{MIN}} \text{ and } j \in N^+(i) \\ p_i(j) \text{ if } i \in V_R \\ 0 \text{ otherwise} \end{cases}$$

and b the vector of dimension m defined as:

$$b_i = \begin{cases} \sum_{s \in V_S} \sigma(i)(s)\text{Val}(s) \text{ if } i \in V_{\text{MAX}} \\ \sum_{s \in V_S} \tau(i)(s)\text{Val}(s) \text{ if } i \in V_{\text{MIN}} \\ \sum_{s \in V_S} \text{Val}(s)p_i(s) \text{ if } i \in V_R \end{cases}$$

we have the following equality:

$$v_{\sigma,\tau} = Qv_{\sigma,\tau} + b$$

We notice that, since every playable vertices of G has a value greater than zero, there is a path of at most m steps from any vertex to a sink. We recall that $Q^m_{i,j}$ corresponds to the probability of ending in state j starting from i after m steps. Since there is a path of length at most m that reaches a sink, the sum of each row is strictly less than 1. Hence, $||Q^m|| < 1$. Thus, the matrix $(I - Q)$ is invertible, so $v_{\sigma,\tau} = (I - Q)^{-1}b$ and is the only vector that satisfies the condition of Lemma 1.14. Hence, if a vector satisfies the condition of the Lemma, it is the value vector of σ, τ. □

1.3.3 . Finite Game

Let σ and τ be two memoryless strategies. We consider the **operators** $\mathcal{I}$, $\mathcal{I}_\sigma$, $\mathcal{I}_\tau$ and $\mathcal{I}_{\sigma,\tau}$ on the complete lattice $[0,1]^V$ defined below. As we show later, those operators allow us to consider the value of the games that stop after a given number of steps under fixed or optimal strategies.

$$\mathcal{I}_{\sigma,\tau}(v)(x) = \begin{cases} \sum_{y \in N^+(x)} \tau(x)(y)v(y) \text{ if } x \in V_{\text{MIN}} \\ \sum_{y \in N^+(x)} \sigma(x)(y)v(y) \text{ if } x \in V_{\text{MAX}} \\ \sum_{y \in N^+(x)} p_x(y)v(y) \text{ if } x \in V_R \\ \mathsf{Val}(x) \text{ if } x \in V_S \end{cases}$$

$$\mathcal{I}_\sigma(v)(x) = \begin{cases} \min_{y \in N^+(x)} v(y) \text{ if } x \in V_{\text{MIN}} \\ \sum_{y \in N^+(x)} \sigma(x)(y)v(y) \text{ if } x \in V_{\text{MAX}} \\ \sum_{y \in N^+(x)} p_x(y)v(y) \text{ if } x \in V_R \\ \mathsf{Val}(x) \text{ if } x \in V_S \end{cases}$$

$$\mathcal{I}_\tau(v)(x) = \begin{cases} \sum_{y \in N^+(x)} \tau(x)(y)v(y) \text{ if } x \in V_{\text{MIN}} \\ \max_{y \in N^+(x)} v(y) \text{ if } x \in V_{\text{MAX}} \\ \sum_{y \in N^+(x)} p_x(y)v(y) \text{ if } x \in V_R \\ \mathsf{Val}(x) \text{ if } x \in V_S \end{cases}$$

$$\mathcal{I}(v)(x) = \begin{cases} \min_{y \in N^+(x)} v(y) \text{ if } x \in V_{\text{MIN}} \\ \max_{y \in N^+(x)} v(y) \text{ if } x \in V_{\text{MAX}} \\ \sum_{y \in N^+(x)} p_x(y)v(y) \text{ if } x \in V_R \\ \mathsf{Val}(x) \text{ if } x \in V_S \end{cases}$$

We now consider the concept of **finite version** of a game. This method of using finite games to approximate the value of SSGs was already present in the first paper by Shapley [Sha53]. For a game G, the finite game $G^{(i)}$ play similarly as G, but stop after i steps to end in a 0 sink.

Definition 1.15 (Finite Game)**.** *Let $G = (V, E)$ be an SSG. For $i \geq 1$, we construct recursively the games $G^{(i)} = (V^{(i)}, E^{(i)})$ as follows.*

- $V^{(i)} = V \times \{1, \ldots, i\} \cup \{0\}$
- $V_S^{(i)} = V_S \times \{1, \ldots, i\} \cup \{0\}$
- $E^{(1)} = \{((x, 1), 0) \mid \forall x \in V \setminus V_S\}$
- $E^{(i+1)} = E^{(i)} \cup \{((x, i), (y, i-1)) \mid \forall x \in V \setminus V_S,\ (x, y) \in E\}$

The probability distributions are the same as in G and for $s \in V_S$ and $j \leq i$, $Val((s, j)) = Val(s)$ and $Val(0)$ is 0.

We present an example of a finite game in Figure 1.2

Proposition 1.16. *The optimal value vector v of $G^{(i)}$ defined as*

$$v(x) = \inf_\tau \sup_\sigma v_{\sigma,\tau}(x) = \sup_\sigma \inf_\tau v_{\sigma,\tau}(x)$$

exists and is equal for $1 \leq j \leq i$ to:

$$v((x, j)) = \mathcal{I}^j(\boldsymbol{0})(x)$$

where $\boldsymbol{0}$ is the null vector.

Proof. For i a fixed integer, this is direct by finite induction on j since $G^{(i)}$ is an acyclic game. □

Every strategy on G can be directly interpreted as a strategy on $G^{(i)}$. For simplification, we also identify vertices x of G and vertices (x, i) of $G^{(i)}$.

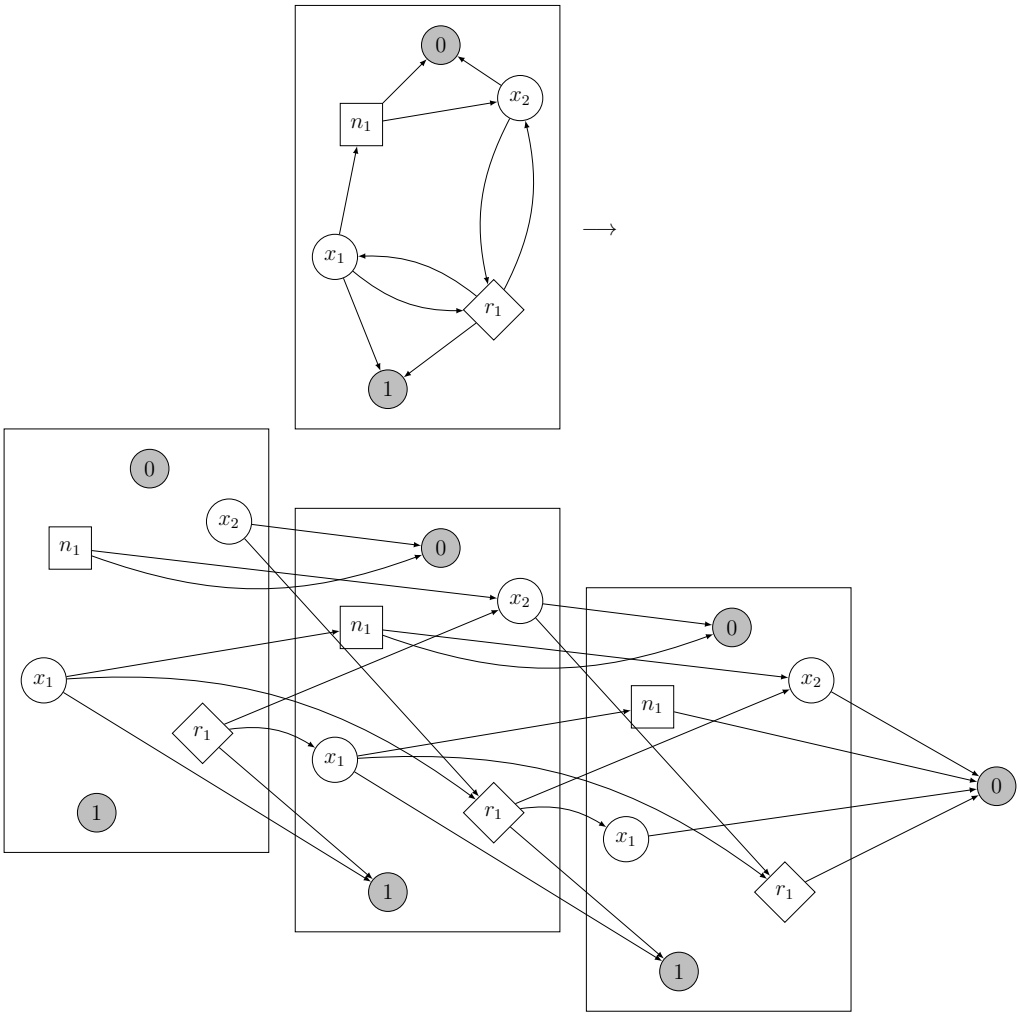


Figure 1.2: Transformation of the game G into the finite game $G^{(3)}$

Corollary 1.17. *Let σ, τ be two memoryless strategies of G. The value of game $G^{(i)}$ under strategies σ, τ is given for $1 \leq j \leq i$ by:*

$$v_{\sigma,\tau}((x,j)) = \mathcal{I}^j_{\sigma,\tau}(\mathbf{0})(x)$$

There is a pure best response $\tau(\sigma)$ to σ in $G^{(i)}$ and the value of game $G^{(i)}$ under strategy $\sigma, \tau(\sigma)$ is given for $1 \leq j \leq i$ by:

$$v_{\sigma,\tau}((x,j)) = \mathcal{I}^j_{\sigma}(\mathbf{0})(x)$$

There is a pure best response $\sigma(\tau)$ to τ in $G^{(i)}$ and the value of game $G^{(i)}$ under strategy $\sigma(\tau), \tau$ is given for $1 \leq j \leq i$ by:

$$v_{\sigma,\tau}((x,j)) = \mathcal{I}^j_{\tau}(\mathbf{0})(x)$$

Proof. By induction on j, this is direct using the fact that $G^{(i)}$ is acyclic and Bayes's formula. □

If the game G is stopping, then $\lim_{k \to +\infty} \mathcal{I}^k(\mathbf{0})$ converges towards the optimal value vector of G. We later show that this also holds when G is not stopping.

1.3.4 . SSG with One Player

In order to prove that memoryless strategies admit positional best response, we focus on the one-player version of SSGs. This is equivalent to the **reachability condition** for Markov Decision Process, which has been introduced by Bellman in [Bel57]. We will use similar proofs as the one provided in the chapter of Markov Decision Process of [FBB+21].

Proposition 1.18. *Let G be an SSG with $V_{\text{MIN}} = \emptyset$. There exists an* **optimal positional strategy** *σ^*.*

Proof. We recognise a Markov Decision Process. Since the game is memoryless with perfect information, the computation of the value vector does not depend on the history. Hence, when in a MAX vertex, player MAX has to play as if it was starting in this vertex, and we can restrict ourselves to the study of memoryless strategies. Let v^* be the optimal value vector, defined as:

$$v^*(x) = \sup_{\sigma \in \Sigma_R^{\text{MAX}}} v_\sigma(x)$$

for every vertex x, the value $v^*(x)$ is reached for some strategy σ^x by compacity of Σ_R^{MAX}. The vector v^* satisfies the following Bellman equation [Bel57]:

$$v^*(x) = \max_{y \in N^+(x)} v^*(y) \text{ if } x \in V_{\text{MAX}}$$

$$v^*(x) = \sum_{y \in N^+(x)} p_x(y) v^*(y) \text{ if } x \in V_R$$

$$v^*(s) = \text{Val}(s) \text{ if } s \in V_S$$

$$v^*(x) = 0 \text{ if } x \in K^G$$

Indeed, for the sake of contradiction, let suppose that for some $x \in V_{\text{MAX}}$, $v^*(x) \neq \max_{y \in N^+(x)} v^*(y)$. We call $x^* = \arg\max_{y \in N^+(x)} v(y)$ this would imply that $v^*(x) < v^*(x^*)$. Notice that for every vertex y

reachable from x under σ^x, $v_{\sigma^x}(y) = v^*(y)$. Otherwise, the pure strategy that switches to strategy σ^y when reaching y ensures a better value, which contradicts the optimality of σ^x for game starting in x. Let y with $\sigma^x(x)(y) > 0$ and $v^*(y) < v^*(x^*)$. Such a y exist since $v^*(x) < v^*(x^*)$. We consider σ' the memoryless strategy that is identical to σ^x, except that $\sigma'(x)(y) = \frac{1}{2}\sigma^x(x)(y)$ and $\sigma'(x)(x^*) = \sigma^x(x)(x^*) + \frac{1}{2}\sigma^x(x)(y)$. This does not increase the size of K_{σ^x}, hence by Lemma 1.14, $v_{\sigma'}(x) > v_{\sigma^x}(x)$. Therefore, v^* satisfies the given Bellman equation.

Now we need to prove that there is a MAX positional strategy σ^* that satisfies v^*. We consider the game G', which is a copy of G where the only edges that have been kept are the one outgoing from random vertices and the edges (x, y) with $x \in V_{\text{MAX}}$ and $v^*(x) = v^*(y)$. We consider σ^* a positional positive MAX strategy which can be computed in polynomial time (see Subsection 1.3.1 for more details). We now show that σ^* is an optimal strategy in G. To do so, we notice that by construction of v^*, for all vertex x such that $v^*(x) > 0$ there is a path from x to a sink s, $(x, x_1, \ldots, x_k, s)$ such that $v^*(x) \leq v^*(x_1) \leq \ldots \leq v^*(x_k) \leq v(s)$. By definition of G', this path also exists in G'. Hence $K^{G'} = K^G$, therefore, by Lemma 1.14 $v^G_\sigma = v^G_\sigma = v^*$ and there exists an optimal positional MAX strategy. □

Proposition 1.19 ([Con92, FBB+21])**.** *Let G be an SSG with $V_{\text{MIN}} = \emptyset$. An optimal positional* MAX *strategy can be computed in polynomial time.*

Proof. The proof of Proposition 1.18 shows that an optimal positional MAX strategy can be computed in $O(|E|)$ if the optimal value vector is given. Let G be a SSG with no MIN vertices. We prove that the optimal value vector v^* of G can be found by solving a linear programming problem. We consider $\mathcal{L}$ the following linear program.

$$
\begin{aligned}
\text{Minimise } & \sum_{x \in V} v(x) & & \\
\text{Subject to } & v(s) = \text{Val}(s) & & \text{for } s \in V_S \\
& v(x) \geq v(y) & & \text{for } x \in V_{\text{MAX}} \text{ and } y \in N^+(x) \\
& v(x) \geq \sum_{y \in N^+(x)} p_x(y) v(y) & & \text{for } x \in V_R \\
& v(x) \geq 0 & & \text{for } x \in V
\end{aligned}
$$

Recall the operator $\mathcal{I}$. By Proposition 1.16, $\mathcal{I}^k(\mathbf{0})$, where $\mathbf{0}$ is the null vector, corresponds to the optimal value vector of $G^{(k)}$. Since the operator $\mathcal{I}$ is monotone, $\lim \mathcal{I}^k(\mathbf{0})$ converges towards the least fixed point of $\mathcal{I}$. Moreover, since there exists a memoryless optimal strategy of G, under that strategy, the probability of the game lasting more than k steps before ending in a sink tends to 0. Hence, the optimal value vector of $G^{(k)}$ converges towards the optimal value of G. Therefore, v^* is equal to $\lim \mathcal{I}^k(\mathbf{0})$. Let v be a feasible solution of $\mathcal{L}$. We notice that $\mathcal{I}(v) \leq v$ and $\mathcal{I}(v)$ is also a feasible solution of $\mathcal{L}$. Hence, $v' = \lim_{k \to +\infty} \mathcal{I}^k(v) \leq v$, which exists by compacity, is also a feasible solution of $\mathcal{L}$ and since v^* is the least fixed point of $\mathcal{I}$, we have $v^* \leq v' \leq v$. Hence, v^* is the unique optimal solution of $\mathcal{L}$. We recall that linear programs can be solved in polynomial time in $O\left(m^\omega \log(p)\right)$ with ω the exponent in the complexity of matrix multiplication, m the size of $\mathcal{L}$ and p the precision [JSWZ20]. □

Corollary 1.20. *Let G be an SSG such that $V_{\text{MAX}} = \emptyset$, G admits an optimal* MIN *positional strategy that can be computed in polynomial time.*

Proof. The proof is similar to the MAX case, with the exception that K^G needs to be computed and replaced with sinks of value 0 before expressing the linear program. □

1.3.5 . Best Response to Memoryless Strategies

We notice that for a fixed memoryless MAX strategy σ, finding a best response to σ corresponds to finding an optimal MIN strategy to the one-player game where every MAX node x has been replaced by a random vertex with probability distribution $\sigma(x)$.

Proposition 1.21. *Let σ be a memoryless* MAX *strategy, there exists a positional best response $\tau(\sigma)$ that can be computed in polynomial time.*

Let τ be a memoryless MIN *strategy, there exists a positional best response $\sigma(\tau)$ that can be computed in polynomial time.*

Proof. This is direct from Proposition 1.18 and Corollary 1.20 □

For a given pair of positional MAX and MIN strategy, (σ, τ) it is possible to know if τ is in $BR(\sigma)$, i.e. τ is a positional best response to σ, by computing $v_{\sigma,\tau}$.

Lemma 1.22. *Given positional strategies (σ, τ) and a real $|V|$-dimensional vector v, one has equality between v and $v_{\sigma,\tau}$ if and only if the following conditions are met:*

(i) *For $s \in V_S$, $v(s) = Val(s)$*

(ii) *For $r \in V_R$, $v(r) = \sum_{y \in N^+(r)} p_r(y)v(y)$*

(iii) *For $x \in V_{\text{MIN}}$, $v(x) = v(\tau(x))$*

(iv) *For $x \in V_{\text{MAX}}$, $v(x) = v(\sigma(x))$*

(v) *For any $x \in V$, $v(x) = 0$, if and only if $x \in K^G_{\sigma,\tau}$*

Moreover, $\tau \in BR(\sigma)$ if and only if for any x in V_{MIN}, $v(x) = \min_{y \in N^+(x)} v(y) = v(\tau(x))$ and the last condition is modified into $v(x) = 0$ if and only if $x \in K^G_\sigma$.

Proof. The first part is a reformulation in the positional case of Lemma 1.14. For σ a positional MAX strategy and $\tau \in BR(\sigma)$, τ is the optimal strategy in the game where every MAX vertex x has been replaced by a random vertex with same probability distribution as σ, so in the case of positional strategy probability 1 to go to $\sigma(x)$. Hence, τ satisfies the Bellman equation that we presented in the proof of Proposition 1.18 and $v_{\sigma,\tau}$ satisfy the given condition.

In order to prove the converse, we will show that if a positional strategy τ satisfies the condition of Lemma 1.22, then for every positional strategy τ', $v_{\sigma,\tau} \leq v_{\sigma,\tau'}$.

First, we replace every vertex of K_σ with a sink of value 0 and every MAX vertex with a random vertex with same probability distribution. Then, we consider the operator $\mathcal{I}_{\tau'}$. Similarly, as for

Corollary 1.17, for $k \geq 0$, and any vertex v, the value of $\mathcal{I}^k_{\tau'}(v)$ corresponds to the game where player MIN plays as τ' during k steps to reach some vertex x and if x is a sink, MIN pays $\mathrm{Val}(x)$ and otherwise, MIN pays $v(x)$. Since, $v_{\sigma,\tau}(x) = \min_{y \in N^+(x)} v(y)$, it results that $\mathcal{I}_{\tau'}(v_{\sigma,\tau}) \geq v_{\sigma,\tau}$. Moreover, $\mathcal{I}^{k+1}_{\tau'}(v_{\sigma,\tau}) \geq \mathcal{I}^k_{\tau'}(v_{\sigma,\tau})$. Hence, $v' = \lim_{k \to +\infty} \mathcal{I}^k_{\tau'}(v_{\sigma,\tau}) \geq \mathcal{I}_{\tau'}(v_{\sigma,\tau}) \geq v_{\sigma,\tau}$. Since the transformed game ends with probability 1, v' is the value of the game where all vertices of K_σ have been replaced with sinks 0, under strategy (σ, τ'). Hence, $v' \geq v_{\sigma,\tau}$ and it results that $v_{\sigma,\tau} \leq v_{\sigma,\tau'}$. Hence, τ is a positional best response to σ. □

Definition 1.23 (Value of a Strategy)**.** *For σ a positional* MAX *strategy, the value vector of σ noted v_σ is the value of $v_{\sigma,\tau(\sigma)}$ where $\tau(\sigma)$ is a positional best response to σ.*

Similarly, the value vector of τ a MIN strategy, noted v_τ, is the value of $v_{\sigma(\tau),\tau}$ where $\sigma(\tau)$ is a positional best response to τ.

1.4 . SSG with Two Players

In this section, we show that for any SSG G, there exists a pair of positional strategies such that their value vector is the optimal value vector of G. The proof given is adapted from the one in [FBB+21].

We consider the operator $\mathcal{I}$ on the complete lattice $[0,1]^V$, and we recall that since it is monotonic it admits a least fixed point, that we denote by v^*.

Lemma 1.24. *For all $x \in V$,*

$$v^*(x) \leq \sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} \inf_{\tau \in \Sigma^{\text{MAX}}_{gen}} v_{\sigma,\tau}(x) \leq \inf_{\tau \in \Sigma^{\text{MAX}}_{gen}} \sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} v_{\sigma,\tau}(x)$$

Proof. Let x be a vertex of V. The rightmost inequality is classic and is not specific to SSGs.

$$\begin{aligned}
\forall \sigma', \forall \tau', \ v_{\sigma',\tau'}(x) &\leq \sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} v_{\sigma,\tau'}(x) \\
\forall \sigma', \forall \tau', \ \inf_{\tau \in \Sigma^{\text{MIN}}_{gen}} v_{\sigma',\tau'}(x) &\leq \sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} v_{\sigma,\tau'}(x) \\
\forall \tau', \ \sup_{\sigma' \in \Sigma^{\text{MAX}}_{gen}} \inf_{\tau \in \Sigma^{\text{MIN}}_{gen}} v_{\sigma',\tau}(x) &\leq \sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} v_{\sigma,\tau'}(x) \\
\sup_{\sigma' \in \Sigma^{\text{MAX}}_{gen}} \inf_{\tau \in \Sigma^{\text{MIN}}_{gen}} v_{\sigma',\tau}(x) &\leq \inf_{\tau' \in \Sigma^{\text{MIN}}_{gen}} \sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} v_{\sigma,\tau'}(x)
\end{aligned}$$

Let us prove that $v(x) = \sup_{\sigma' \in \Sigma^{\text{MAX}}_{gen}} \inf_{\tau \in \Sigma^{\text{MIN}}_{gen}} v_{\sigma',\tau}(x)$ is a fixed point of $\mathcal{I}$.

In the next steps of the proof, for readability reasons, we will not specify the set of strategies, which will be $\Sigma^{\text{MAX}}_{gen}$ for MAX and $\Sigma^{\text{MIN}}_{gen}$ for MIN.

We now prove that v is a fixed point of $\mathcal{I}$. First, for any vertex $x \in V_{\text{MAX}}$, we recall that for $h \in V^*$ $\sigma(h)(y)$ is a probability distribution corresponding to the probability to go to y after having seen history h and $v_{\sigma,\tau}(h)$ corresponds to the expected gain of MAX following strategies (σ, τ) after seeing history h.

$$
\begin{aligned}
v(x) &= \sup_{\sigma} \inf_{\tau} \sum_{y \in N^+(x)} \sigma(x)(y) v_{\sigma,\tau}(xy) \\
&= \sup_{\sigma} \sum_{y \in N^+(x)} \sigma(x)(y) \inf_{\tau} v_{\sigma,\tau}(xy) \\
&= \max_{y \in N^+(x)} \sup_{\sigma \atop \sigma(x)(y)=1} \inf_{\tau} v_{\sigma,\tau}(xy) && \text{since } \sigma(x) \text{ is a probability distribution} \\
&= \max_{y \in N^+(x)} \sup_{\sigma'} \inf_{\tau} v_{\sigma',\tau}(y) && \text{since the game is memoryless} \\
&= \mathcal{I}(v)(x)
\end{aligned}
$$

We proceed similarly for the vertices of V_{MIN} and V_R. Since, v is a fixed point of $\mathcal{I}$ it results that $v \geq v^*$. □

Lemma 1.25. *For all $x \in V$,*

$$v^*(x) \geq \inf_{\tau \in \Sigma^{\text{MAX}}_{gen}} \sup_{\sigma \in \Sigma^{\text{MAX}}_{gen}} v_{\sigma,\tau}(x)$$

Proof. We consider τ^* a positional strategy computed in the following way: for every vertex x of V_{MIN} define $\tau^*(x) = y$ with $y \in N^+(x)$ such that $v^*(x) = v^*(y) = \min_{z \in N^+(x)} v^*(z)$. Such a y exists since v^* is a fixed point of $\mathcal{I}$. With τ^* fixed, we can compute $\sigma(\tau^*)$ a positional best response to τ^*. We know that $v_{\sigma(\tau^*),\tau^*}$ is the least fixed point of $\mathcal{I}_{\tau^*}$ by proof of Corollary 1.20. Moreover, v^* is also a fixed point of $\mathcal{I}_{\tau^*}$. Hence,

$$v^* \geq v_{\sigma(\tau^*),\tau^*} = \sup_{\sigma} v_{\sigma,\tau^*} \geq \inf_{\tau} \sup_{\sigma} v_{\sigma,\tau}.$$

□

Theorem 1.26. *Let G be an SSG, there is a pair of positional strategies σ^*, τ^* such that v_{σ^*,τ^*} is the optimal value vector of G.*

Proof. From Lemma 1.24 and 1.25, we can conclude that v^* is the optimal value vector of G. The proof of Lemma 1.25 shows a pair of positional strategies(σ^*, τ^*) such that $v^* = v_{\sigma^*,\tau^*}$. □

Notice that it is not necessarily true with our choice of σ^* that v_{σ^*} equals to v^*. In order to find a pair of optimal strategies in which both strategies are a best response of the other, we give the following proposition.

Proposition 1.27. *Given v^* the optimal value vector of a game G,* **a pair of optimal positional strategies** (σ^*, τ^*) *can be computed in polynomial time, such that $v_{\sigma^*} = v_{\tau^*} = v_{\sigma^*,\tau^*} = v^*$ or, in other words, $\sigma^* \in BR(\tau^*)$ and $\tau^* \in BR(\sigma^*)$.*

Proof. We use a similar technique as the one in the proof of Proposition 1.18. We consider the game G' where we kept only the edges (x, y) from playable vertices such that $v^*(x) = v^*(y)$. We then compute a positive positional strategy σ^* of G', and we consider any MIN strategy τ^* of G'. Then, $K^G_{\sigma^*,\tau^*} = K^G$ and by Lemma 1.22, τ^* is a best response to σ^*. The proof of Lemma 1.25 ensures that σ^* is a best response to τ^*, which conclude the proof. □

Lemma 1.28 (Optimality Condition)**.** *Let σ an τ a pair of optimal strategies with value vector $v = v_{\sigma,\tau}$. The pair (σ, τ) is a pair of optimal strategies if and only if:*

- $v(x) = \max_{y \in N^+(x)} v(y)$ *if* $x \in V_{\text{MAX}}$
- $v(x) = \min_{y \in N^+(x)} v(y)$ *if* $x \in V_{\text{MIN}}$
- $v(x) = \sum_{y \in N^+(x)} p_x(y)v(y)$ *if* $x \in V_R$
- $x \in K_\sigma^G$ *if and only if* $v(x) = 0$

Proof. The optimal value vector v^* satisfies those conditions, thus, by definition, if σ and τ are optimal, their value vector satisfies those optimality conditions.

If $v_{\sigma,\tau}$ satisfies the condition stated above, then by Lemma 1.22, τ is a best response to σ and similarly to the proof of Proposition 1.18, we can show that σ is a best response to τ. Hence, for all x:

$$v(x) = \inf_\tau v_{\sigma,\tau}(x) \leq \sup_{\sigma'} \inf_\tau v_{\sigma',\tau}(x) = v^*(x)$$

Thus, $v \leq v^*$. However, v^* is the least fixed point of $\mathcal{I}$ and v is a fixed point of $\mathcal{I}$, thus $v^* = v$. □

1.5 . Relation with other games

In this section, we present different games and explain how they relate to each others.

1.5.1 . Deterministic Games

Parity Games

Parity Games were introduced under this name by Emerson and Jutla in [EJ91]. For more detailed information on parity games, one can read [FBB+21, GT02].

Definition 1.29. *A Parity Game is a directed graph $G = (V, E)$, together with:*

1. *A partition of the vertex set V into two parts: V_1 and V_2.*
2. *All vertices with out-degree at least one.*
3. *A set $\mathcal{P} = \{1, \ldots, d\}$ of priorities.*
4. *A function γ from V to $\mathcal{P}$.*

The game is played in turn as for SSGs: there are two players, 1 and 2 that move a token alongside the arcs of the graph. If the token is in V_i, then player i moves the token. A play is an infinite word w in V^ω that is coherent with the edges of the game. $\gamma(w)$ gives a word in $\mathcal{P}^\omega$. We say that player 1 wins the game *if the largest priority that appears infinitely often in $\gamma(w)$ is even.*

This game is deterministic, and there exists a pair of optimal strategies that are positional [Mos91, EJ91]. In 2017, Calude, Jain, Khoussainov, Li, and Stephan presented a **quasipolynomial time algorithm** for solving parity games [CJK+20]. One of the interests in studying this type of game is that solving parity game is linear-time equivalent to the model-checking problem for modal μ-calculus [EJS93].

Mean Payoff and Discounted Payoff Games

Mean payoff and **discounted payoff** games are similar to parity games with different objectives.

Definition 1.30 (Mean Payoff Games [EM79]). *In mean payoff games, player* 1 *wants to maximise* $\limsup_k \frac{1}{k}\sum_{i=0}^{k}\gamma(w)_i$. *The goal of player* 2 *is to minimise this value.*

The strategies of mean payoff games are positional [EM79]. Moreover, mean payoff games can be solved in **pseudo-polynomial time** as Zwick and Paterson showed in [ZP96]. In the same paper, they introduced Discounted Games.

Definition 1.31 (Discounted Games [ZP96]). *In discounted games, we consider a discount factor* $\lambda \in (0,1)$. *The objective of player* 1 *is to maximise the value*

$$(1-\lambda)\sum_{i=0}^{+\infty}\lambda^i\gamma(w)_i$$

while player 2 *wants to minimise it.*

Here, $(1-\lambda)$ is just a normalisation factor.

Reductions

In [Pur97], Puri presents a reduction from parity games to mean-payoff games. The reduction is as follows: the game is unchanged, but the rewards function of the mean payoff games is defined as r with: $r(v) = (-|V|)^{\gamma(v)}$.

In [ZP96], Zwick and Paterson gives the following reduction from mean payoff to discounted payoff games. The game is kept unchanged, and the considered discount factor is:

$$\beta = 1 - \frac{1}{4|V|^3 \max_{v\in V}|r(v)|}$$

They also provide in the same paper a reduction from discounted payoff games to simple stochastic games. Let $G = (V, E)$ be a discounted payoff games with V partitioned in V_1 and V_2, with reward function r and discounted factor β. Let $l = \min_{v\in V} r(v)$, $t = \max_{v\in V} r(v)$ and $d = \max(1, u - l)$.

We consider the following SSG $G' = (V', E')$ with:

$$V' = V_{\text{MAX}} \cup V_{\text{MIN}} \cup V_R \cup V_S$$

$$V_{\text{MAX}} = V_1,\ V_{\text{MIN}} = V_2,\ V_R = E,\ V_S = \{0,1\}$$

$$E' = \{(x,(x,y)), ((x,y),y), ((x,y),0), ((x,y),1) \mid x \in V, (x,y) \in E\}$$

$$p_{(x,y)}(z) = \begin{cases} \beta \text{ if } z = y \\ (1-\beta)\dfrac{r(v)-l}{d} \text{ if } z = 1 \\ (1-\beta)\left(1 - \dfrac{r(v)-l}{d}\right) \text{ if } z = 1 \end{cases}$$

The value of G corresponds to the value of G' by an affine transformation. Hence, the order between the strategies are preserved.

A direct reduction from parity games to Simple Stochastic Games has been provided by Chatterjee and Fijalkow in [CF11].

1.5.2 . One Player Games

As stated in Subsection 1.3.4, an SSG with no MIN vertex (or no MAX vertex) is a **Markov Decision Process** (MDP). To be more precise, with no MIN vertices, it is a Markov Decision Process with a reachability objective. One can also consider SSG with other objectives, such as mean payoff and discounted payoff [Bel57]. A value $r(x)$ is associated to each vertex x of the game, and the player wants to maximise an objective. For a play π, the value $\limsup_k \frac{1}{k} \sum_{i=0}^{k} r(\pi_i)$ for mean payoff MDPs and $\sum_{k=0}^{+\infty} \beta^i r(\pi_i)$ for discounted payoff MDP with discount factor $\beta \in (0,1)$. All those versions can be solved in polynomial time using linear programming with similar proof as in Proposition 1.19.

1.5.3 . Equivalence with other Stochastic Games

In [AM09], Anderson and Milterson showed that the problem of solving the following stochastic games are polynomial-time equivalent.

- Stochastic parity games. Here, we add random vertices to parity games that work in the same way as SSG.
- Stochastic Mean-Payoff Games with rewards and probabilities given in unary.
- Stochastic Mean-Payoff Games with rewards and probabilities given in binary.
- Stochastic Discounted-Payoff Games with rewards and probabilities given in binary.

2 - Value Iteration Algorithm and Quadratic Programming

Contents

2.1 . Values Format of the Nodes

For every pair of positional strategies, the values of the vertices are rational numbers. We give a description of the value of a q-SSG parametrised by its number of random vertices. We recall that we denote by V_R the set of random vertices and $r = |V_R|$. This allows us to know at which precision to approximate the value vector of an SSG. We have presented this work in [ABdMS21]. In a q-SSG, there is a function $f(q, r)$ such that, for every pair of positional strategies (σ, τ), there exists $t \leq f(q, r)$, such that for every vertex x, there is an integer p_x, such that $v_{\sigma,\tau}(x) = \frac{p_x}{t}$

Condon proved in [Con92] that $f(2, r) \leq 4^r$. Then Auger, Coucheney and Strozecki improved this to $f(2, r) \leq 6^{r/2}$ in [ACS14]. We show that $f(q, r) = q^r$ for q-SSGs, which gives the improved bound of $f(2, r) \leq 2^r$ for 2-SSGs.

Theorem 2.1. *Let $q \geq 1$ and G a q-SSG with r random vertices, then for any pair of strategies (σ, τ) there is $t \leq q^r$ such that, for every vertex x, there is an integer p_x such that, $v_{\sigma,\tau} = \frac{p_x}{t}$.*

Proof of Theorem 2.1 relies on the matrix tree theorem applied to a directed multigraph representing the game under a pair of strategies. A similar result on Markov chains has been independently proven by Skomra in [Sko21] using the Markov chain tree formula.

Let us show that q^r is a tight bound for $f(q, r)$. Consider a Markov chain (an SSG with no MAX nor MIN vertices) with $r + 2$ vertices: two sinks 0 and 1 and r random vertices $x_1, \ldots, x_r$. Vertex x_1 goes to 1 with probability $1/q$ and to 0 with probability $(q - 1)/q$. For $r \geq i \geq 2$, x_i goes to 0 with probability $(q - 1)/q$ and to x_{i-1} with probability $1/q$. Then, the value of x_r is q^{-r}.

Let us remark that a q-SSG can be assumed to have all its probability transitions of the form p/q. The idea here is to notice that it is possible to loop with a certain probability on the same random vertex without changing the values.

Lemma 2.2. *Let G be a q-SSG, then there is G' a q-SSG with the same vertices and same outneighbourhood for playable vertices such that for every pair of strategies (σ, τ), $v^G_{\sigma,\tau} = v^{G'}_{\sigma,\tau}$ and such that for all $x \in V_R$ and all $x' \in N^+(x)$ there is an integer $p_{x,x'}$ such that $p_x(x') = p_{x,x'}/q$.*

Proof. For a a random vertex in G, and $q_a < q$ such that for every other vertex x in G there is $p_x \in \mathbb{N}$ and a probability p_x/q_a to go directly from a to x, we change in G' those probabilities to p_x/q and we add a probability p/q to stay in a, where:

$$p = q - \sum_{x \in V} p_x.$$

We suppose that $p_a = 0$. The probability for $x' \in N^+(x)$ to be the first vertices different of x to be reached is thus

$$\frac{p_x}{\sum\limits_{y \in N^+(x)} p_y} = \frac{p_x}{q_a}$$

Hence, for σ and τ a pair of strategies, $v^G_{\sigma,\tau} = v^{G'}_{\sigma,\tau}$. □

Now, we state the classical matrix-tree theorem that we use in our proof (see e.g. [CK78]). Let G be a directed multigraph with n vertices, then the *Laplacian matrix* of $\mathcal{G}$ is a $n \times n$ matrix $L(\mathcal{G}) = (l_{i,j})_{i,j \leq n}$ defined by:

(i) $l_{i,j}$ equals $-m$ where m is the number of arcs from i to j.

(ii) $l_{i,i}$ is the number of arcs entering i, excluding self-loops.

Theorem 2.3 (Matrix tree theorem for directed multigraphs)**.** *For $G = (V, E)$ a directed multigraph with vertices $V = \{v_1, \dots, v_k\}$ and L its Laplacian matrix, the number of spanning trees rooted at v_i is $\det(\hat{L}_{i,i})$ where $\hat{L}_{i,i}$ is the matrix obtained by deleting the i-th row and column from L.*

We can now prove Theorem 2.1.

Proof of Theorem 2.1. The beginning of the proof is the same as the one in [Con93] and [ACS14]. We start by transforming the game with fixed strategies in a Markov Chain with equivalent values. Then, we show that the value of each vertex can be written $\dfrac{\det B_i}{\det q(I-A)}$ using Cramer rule, for B_i and A two matrices which will be carefully defined. To conclude, we will show that $\det q(I-A) < q^r$ by creating a graph obtained from our initial game and using Theorem 2.3.

We consider a q-SSG G and two positional strategies σ and τ. Without loss of generality, we can restrict ourselves to the computation of non-zero, non-sink values. Thus, each vertex has a non-zero probability to reach the 1-sink. To compute the values $v_{\sigma,\tau}$, we can consider G_A an SSG with vertex set $V_R \cup V_S$: the random vertices and the sinks of V. The value of the sinks is not changed, and the probability distribution p'_x is defined as follows. For $x \in V_R$ and x' in G_A, we call $M_{x,x'}$ the set of MAX and MIN vertices y in $N^+(x)$ such that there is a path following only arcs of σ and τ from y to x'. We then have

$$p'_x(x') = \sum_{y \in M_{x,x'}} p_x(y)$$

The graph G_A has $r+2$ vertices that we denote by $a_1, \dots, a_{r+1}, a_{r+2}$, where a_{r+1} is the 0-sink and a_{r+2} is the 1-sink. Let b be the r-dimensional column vector with $b_i = p'_{a_i}(a_{r+2})$. We define A the $r \times r$ matrix, with $A_{i,j} = p'_{a_i}(a_j)$.

The values of the random vertices are defined by the vector z that satisfies the following equation:

$$z = Az + b$$

Let I be the identity matrix, $(I - A)$ is invertible because each random vertex has access to a sink, so every eigenvalue of A is strictly less than 1. We refer to [Con93] for details. Hence, the equation has a unique solution and z is also solution of:

$$q(I - A)z = qb$$

Hence, under the strategies σ, τ, the value z_i of a random vertex a_i given by the Cramer rule is

$$z_i = \frac{\det B_i}{\det q(I-A)}$$

where B_i is the matrix $q(I - A)$ where the i-th column has been replaced by qb. The value $\det B_i$ is an integer. See [ACS14] for more details. Our goal is now to bound $\det q(I - A)$.

From the graph G_A, we construct the graph G' by inverting all arcs, and duplicating an arc of probability p/q into p arcs of probability $1/q$. We also add an arc coming from the 1-sink to the 0-sink

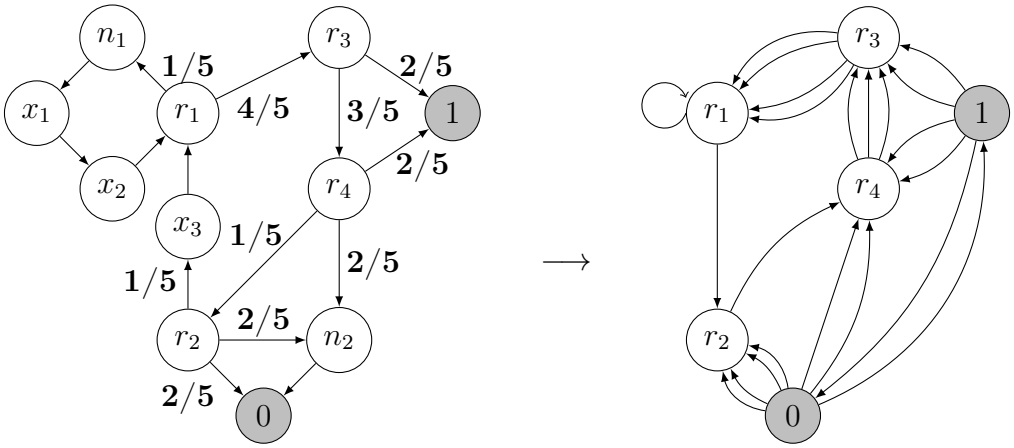


Figure 2.1: Example of a transformation of a graph G into a graph G'

and one from the 0-sink towards the 1-sink. Figure 2.1 shows an example of the transformation from G to G'. The Laplacian L of G' is thus the following matrix.

$$L = \left(\begin{array}{c|cc} q(I-A)^T & B \\ \hline 0 & \begin{array}{cc} 1 & 1 \\ 1 & 1 \end{array} \end{array}\right)$$

Indeed, every random vertex has in-degree q minus the number of loops. Thus, the number of spanning trees of G' rooted in the 1-sink is equal by Theorem 2.3 to $\det \hat{L}_{r+2,r+2}$ where we have

$$\hat{L}_{r+2,r+2} = \left(\begin{array}{c|c} q(I-A)^T & B' \\ \hline 0 & 1 \end{array}\right).$$

In other words, the number of spanning trees of G' is equal to $\det q(I-A)$. Furthermore, each spanning tree contains exactly one incoming arcs for every random vertices, and the arc (a_{r+2}, a_{r+1}) has to be used. Thus, there are at most q^r spanning trees rooted in G' and:

$$\det q(I-A) \leq q^r.$$

□

2.2 . Transforming SSG into Stopping SSG

It is well known that any general SSG can be approximated by a stopping SSG. In [Con92], Condon presents a transformation of the game G in G' such that for v and v' the optimal value vector, for any $x \in V$, $v'(x) > 0.5$ if and only if $v(x) > 0.5$. This notion has been introduced to simplify the proof of some properties, but we show in this thesis that this condition can generally be removed. We use the same transformation and a similar proof to show that every SSG can be approximated by a stopping SSG.

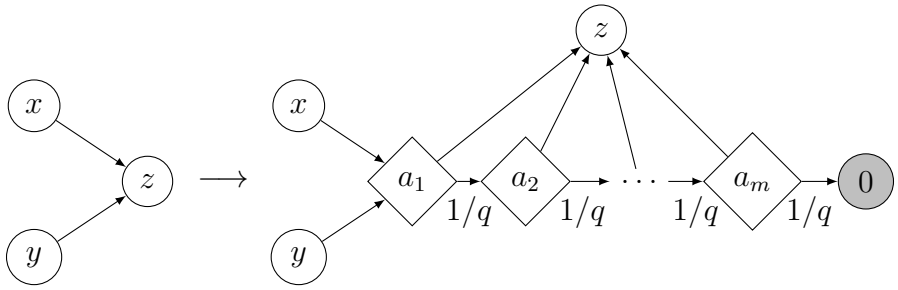


Figure 2.2: Transformation of an edge to obtain a $1/q^m$-transformation. The edges from random vertices a_i to y have probability $(q-1)/q$.

Definition 2.4 (ϵ-transformation)**.** *For $0 < \epsilon < 1$ the ϵ transformation of an SSG $G = (V, E)$ is the game $G_\epsilon = (V', E')$ with*

$$V' = V \cup \{y' \mid y \in V\}$$

where for all $y \in V$, y' is a random vertex.

$$E' = \{(x, y') \mid (x, y) \in E\} \cup \{(y', y) \mid (x, y) \in E\} \cup \{(y', 0) \mid y \in V\}$$

with probability distribution, $p_{y'}(y) = 1 - \epsilon$ and $p_{y'}(0) = \epsilon$.

Thus, G_ϵ is the game where at each turn, there is a probability ϵ to end the game by going to a 0 sink. We notice that for a q-SSG, a $1/q^m$ transformation can be done by adding m successive random vertices before each vertex, as shown in Figure 2.2. Hence, a $1/q^m$ transformation of a game G has $(m+1)|V|$ nodes with $m|V| + r$ random nodes.

The strategies of G are in bijection with the strategies of G_ϵ. We now prove that for ϵ small enough, for any pair of positional strategies, $v^G_{\sigma,\tau}$ is close to $v^{G_\epsilon}_{\sigma,\tau}$. In their original proof, Condon gave a bound for n large enough [Con92]. We give an explicit bound on ϵ with no condition on n.

Proposition 2.5. *For $\epsilon < q^{-3r}/(2|V|^2)$, the difference between vector value of G and G' under fixed strategies (σ, τ) is less than q^{-r}.*

Proof. Let G be an SSG and G' be an ϵ transformation of G. Let σ and τ be two positional strategies.We notice that the vertices of $K^G_{\sigma,\tau}$ are also vertices of $K^{G'}_{\sigma,\tau}$. Hence, if we replace every vertex of $K^G_{\sigma,\tau}$ by a sink of value 0, by Lemma 1.22 the value of G and G' are not modified. We can assume that G under strategies (σ, τ) ends with probability 1, and we recall that by construction G' is also stopping. We write $m = |V \setminus V_S|$, $v = v^G_{\sigma,\tau}$ and $v' = v^{G'}_{\sigma,\tau}$. In this proof, we consider the infinity norm.

We define Q the matrix of dimension $m \times m$. defined as:

$$Q_{i,j} = \begin{cases} 1 \text{ if } i \in V_{\text{MAX}} \text{ and } \sigma(i) = j \\ 1 \text{ if } i \in V_{\text{MIN}} \text{ and } \tau(i) = j \\ p_i(j) \text{ if } i \in V_R \\ 0 \text{ otherwise} \end{cases}$$

and let b the vector of dimension m defined as:

$$b_i = \begin{cases} \text{Val}(s) \text{ if } i \in V_{\text{MAX}}, s \in V_S \text{ and } \sigma(i) = s \\ \text{Val}(s) \text{ if } i \in V_{\text{MIN}}, s \in V_S \text{ and } \tau(i) = s \\ \sum_{s \in S} \text{Val}(s) p_i(s) \text{ if } i \in V_R \end{cases}$$

We write $Q' = (1-\epsilon)Q$ and $b' = (1-\epsilon)b$. Thus, $v = Qv + b$ and $v' = Q'v' + b'$. Moreover, since G and G' are stopping, there is a path from any vertex to a sink in less than m steps going through each random vertex at most once. This path is thus followed with probability at least q^{-r}. For each of the r random vertices, there is probability at least $1/q$ to follow a specific arc. The entries of Q^m in (i,j) are equal to the probability to be in j starting from i and following (σ, τ). Thus, $||Q^m|| \leq 1 - (1/q^r)$ and $||Q'^m|| \leq 1 - (1/q^r)$. It results that $(I-Q)$ and $(I-Q')$ are invertible. Thus $v = (I-Q)^{-1}b$ and $v' = (I-Q')^{-1}b'$.

Since, $||Q^k||$ tends to 0, $(I-Q)^{-1} = \sum_{k=0}^{+\infty} Q^k$. Hence, we have:

$$\begin{aligned} v - v' &= \left(\sum_{k=0}^{+\infty} \left(1 - (1-\epsilon)^{k+1}\right) Q^k \right) b \\ ||v - v'|| &\leq \sum_{k=0}^{+\infty} \left(1 - (1-\epsilon)^{k+1}\right) ||Q||^k \\ &\leq \sum_{k=0}^{+\infty} \left(1 - (1-\epsilon)^{k+1}\right) (1 - (1/q^r))^{\lfloor k/m \rfloor} \\ &\leq \sum_{k=0}^{+\infty} \epsilon(k+1) (1 - (1/q^r))^{\lfloor k/m \rfloor} \\ ||v - v'|| &\leq \frac{\epsilon}{1 - (1/q^r)} \sum_{k=0}^{+\infty} (k+1) \left((1 - (1/q^r))^{1/m} \right)^k \end{aligned}$$

Since $q \geq 2$, $\frac{\epsilon}{1-(1/q^r)} \leq 2\epsilon$. Moreover $0 < 1/m < 1$, this implies that $(1-(1/q^r))^{1/m} \leq 1 - (1/mq^r)$. Hence:

$$||v - v'|| \leq 2\epsilon \sum_{k=0}^{+\infty} (k+1) \left(1 - \frac{1}{mq^r}\right)^k$$

We recognise the power series of $\frac{1}{(1-x)^2}$. Thus, we can write:

$$||v - v'|| \leq \frac{2\epsilon}{(1/(mq^r))^2}$$

Hence, we have proven that $||v - v'|| \leq 2\epsilon m^2 q^{2r}$. Thus, for $\epsilon < q^{-3r}/(2m^2)$, the difference of value between v and v' is less than q^{-r} with $v' \leq v$. □

Corollary 2.6. *For G' an ϵ transformation of G with $\epsilon < q^{-3r}/(2m^2)$, the optimal strategies of G' are optimal strategies of G.*

Proof. Let σ^* and τ^* be a pair of optimal strategies of G'. We notice that for every x in V, we have:

$$0 < v^G_{\sigma^*,\tau^*}(x) - v^{G'}_{\sigma^*,\tau^*}(x) < q^{-r}$$

We recall that the distinct values of $v^G_{\sigma^*,\tau^*}(x)$ have a difference of at least q^r by Theorem 2.1. Since under σ^* and τ^* the value in G' satisfies the optimality condition (Lemma 1.28), they are also satisfied in G since $K^G_\sigma = K^{G'}_\sigma$. Thus, σ^*, τ^* are optimal strategies of G. □

It is important to notice that the converse of Corollary 2.6 is not true. For instance, if we consider the game with two MAX vertices x and y and a sink 1 with arcs (x, y), $(x, 1)$, (y, x) and $(y, 1)$. We notice that the strategy $\sigma(x) = y$ and $\sigma(y) = 1$ is optimal in G but not in any ϵ transformation of G.

2.3 . Value Iteration Algorithm

2.3.1 . Value Iteration Algorithm

As stated in [FBB+21], finding the optimal value vector of an SSG consists in finding the least fixed point of the monotonic operator $\mathcal{I}$ on the complete lattice $[0, 1]^V$ defined in Chapter 1.

This implies that $\lim_{k\to+\infty} \mathcal{I}^k(\mathbf{0}) = v^*$ where $\mathbf{0}$ is the null vector and v^* is the optimal value vector. This method was presented by Shapley in their introduction to stochastic games [Sha53]. The idea behind this algorithm is to consider $\mathcal{I}^k(\mathbf{0})$ as the value vector of the Finite Game that stops after k steps. A similar proof is used by Ibsen-Jensen and Miltersen in [IJM12] to study their algorithm that we present in Section 2.3.2.

Lemma 2.7 (Value Domination of the Infinite Game)**.** *For any positional strategy σ and any general strategy τ, $v^{G^{(i)}}_{\sigma,\tau} \leq v^G_{\sigma,\tau}$ and*

$$||v^G_{\sigma,\tau} - v^{G^{(i|V|)}}_{\sigma,\tau}|| \leq (1 - q^{-|V|})^i$$

Proof. First, we transform the game G and $G^{(i)}$ such that any vertex of K^G_σ is replaced by a sink of value 0. This does not change the value of G and $G^{(i)}$. For any pair of general strategy σ and τ and any sink s, we call $\mathbb{P}^x_{\sigma,\tau}(\to s)$ the probability of reaching the sink s starting from x and following strategies σ and τ in G. Let $T^{\sigma,\tau}$ be a random variable defined as the number of steps in a play following (σ, τ) before reaching a sink in G. Note that $T^{\sigma,\tau}$ may be equal to $+\infty$. We can express the vector value in G and $G^{(i)}$ as such

$$v^{G^{(i)}}_{\sigma,\tau}(x) = \mathbb{P}(T^{\sigma,\tau} < i) \sum_{s\in V_S} \mathbb{P}^x_{\sigma,\tau}(\to s \mid T^{\sigma,\tau} < i)\mathrm{Val}(s)$$

$$v^G_{\sigma,\tau}(x) = \sum_{s\in V_S} \mathbb{P}^x_{\sigma,\tau}(\to s)\mathrm{Val}(s)$$

which can also be written using Bayes rules as:

$$\begin{aligned} v^G_{\sigma,\tau}(x) =& \mathbb{P}(T^{\sigma,\tau} < i) \sum_{s\in V_S} \mathbb{P}^x_{\sigma,\tau}(\to s \mid T^{\sigma,\tau} < i)\mathrm{Val}(s) \\ &+ \mathbb{P}(i \leq T^{\sigma,\tau} < +\infty) \sum_{s\in V_S} \mathbb{P}^x_{\sigma,\tau}(\to s \mid +\infty > T^{\sigma,\tau} \geq i)\mathrm{Val}(s) \end{aligned}$$

We notice, that since vertices of K_σ^G has been replaced with 0 sinks, for any general MIN strategy τ, there is a path from any vertex x to a sink. Moreover, each edges of the path have at least probability $1/q$ since G is a q-SSG. Therefore, $\mathbb{P}(i|V| < T^{\sigma,\tau} < +\infty) < (1-q^{-|V|})^i$, which concludes the proof. □

Now we can compute an upper bound on the speed of the convergence of this algorithm.

Proposition 2.8 (Speed of Convergence of Value Iteration)**.** *Let v^* be the optimal value vector of a $q-$SSG G and $N = |V|$. Then we have:*

$$||v^* - \mathcal{I}^{kN}(\boldsymbol{0})|| \leq \left(1 - q^{-N}\right)^k$$

Proof. Let σ^* and τ^* be two optimal strategies of G. Let σ_i^* and τ_i^* be two optimal strategies of $G^{(i)}$. We recall that since we associate strategies of G and $G^{(i)}$, it means that σ_i^* and τ_i^* may not be positional but are pure. We call σ_i' a MAX best response to τ^* in $G^{(i)}$. Thus, we have the following inequalities.

$$\begin{aligned}
v^{G^{(i)}}_{\sigma^*,\tau_i^*} &\leq v^{G^{(i)}}_{\sigma_i^*,\tau_i^*} && \text{since } \sigma_i^* \text{ is a MAX best response to } \tau_i^* \text{ in } G^{(i)}\\
v^{G^{(i)}}_{\sigma_i^*,\tau_i^*} &\leq v^{G^{(i)}}_{\sigma_i^*,\tau^*} && \text{since } \tau_i^* \text{ is a MIN best response to } \sigma_i^* \text{ in } G^{(i)}\\
v^{G^{(i)}}_{\sigma_i^*,\tau^*} &\leq v^{G}_{\sigma_i^*,\tau^*} && \text{by Lemma 2.7}\\
v^{G}_{\sigma_i^*,\tau^*} &\leq v^{G}_{\sigma^*,\tau^*} && \text{since } \sigma^* \text{ is a MAX best response to } \tau^* \text{ in } G\\
v^{G}_{\sigma^*,\tau^*} &\leq v^{G}_{\sigma^*,\tau_i^*} && \text{since } \tau^* \text{ is a MIN best response to } \sigma^* \text{ in } G
\end{aligned}$$

Moreover, by definition of optimal strategy, $v^G_{\sigma^*,\tau^*} = v^*$ and by Proposition 1.16, $v^{G^{(i)}}_{\sigma_i^*,\tau_i^*} = \mathcal{I}(\mathbf{0})$. Hence, we have:

$$v^{G^{(i)}}_{\sigma^*,\tau_i^*} \leq \mathcal{I}(\mathbf{0}) \leq v^* \leq v^G_{\sigma^*,\tau_i^*}$$

Moreover, by Proposition 2.7, we know that $||v^{G^{(kN)}}_{\sigma^*,\tau_{kN}^*} - v^G_{\sigma^*,\tau_{kN}^*}|| \leq (1-q^{-N})^k$. Hence, this concludes the proof. □

Theorem 2.9 (Speed of Value Iteration)**.** *The value iteration algorithm approximate v^* from below by less than q^{-r} in at most $|V|rq^{|V|}\log(q)$ iterations.*

Proof. We recall that for any positional strategies in a q-SSG, there is $t \leq q^r$ such that the value of any vertices under those strategies is p/t for p an integer. Hence, if v^* is approximated from below by less than q^{-r} then, the pair of strategy obtained by a greedy algorithm on the approximated value is a pair of optimal strategies. Thus, we need k iterations such that $(1-q^{-|V|})^k < q^{-r}$.

We want $k\log\left(1-q^{-|V|}\right) < -r\log(q)$.

We recall that for all $0 < x < 1$, $\log(1-x) < -x$. Hence, if $-kq^{-|V|} < -r\log(q)$, then $k\log\left(1-q^{-|V|}\right) < -r\log(q)$. Therefore, the value iteration algorithm stops after $|V|rq^{|V|}\log(q)$ iterations. □

Corollary 2.10. *Let G be an SSG such that starting from any vertices of G the probability of ending in a sink after X steps is p, then the value iteration algorithm needs $Xrp\log(q)$ iterations.*

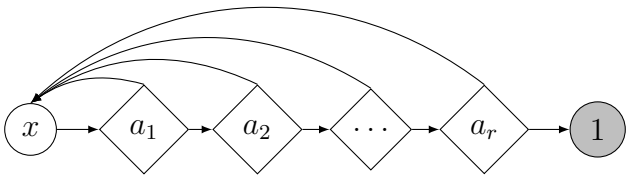


Figure 2.3: An SSG in which the value iteration takes exponential time. Random nodes have uniform probability distribution.

Proof. It is enough to replace $|V|$ with X and $q^{-|V|}$ with p in the previous proofs. □

While $\mathcal{I}^k(\mathbf{0})$ converges towards v^*, it is important to note that approximating v^* can take exponential time. The canonical example of this fact, presented by Condon in [Con93] is the following. Consider the game G with one MIN vertex x, r random nodes $a_1, \ldots, a_r$ and a sink 1. The edges are (x, a_1), for $1 \leq i \leq r-1, (a_i, a_{i+1})$ and (a_i, x), and $(a_r, 1)$, (a_r, x). The probability distributions are uniform. We represent this game in Figure 2.3.

2.3.2 . Ibsen-Jensen and Miltersen Algorithm

In [IJM12], Ibsen-Jensen and Miltersen present a value iteration algorithm for 2-SSG with a sink 1 and a sink 0 that runs in $O(r2^r(r\log(r)+n))$, assuming that the cost of arithmetic operations on number of bit length $\Theta(r)$ is 1. We present this algorithm in Algorithm 2

Algorithm 2: IJMA [IJM12]

Data: G an SSG
Result: v the optimal value vector

```
begin
    v ⟵ (0, ..., 0) of size 2 + |V_R|
    v_1 ⟵ 1
    for i ∈ {1, ..., T} do
        v ⟵SolveDGG(G, v)
        v' ⟵ v
        for x ∈ V_R with neighbours y and z do
            v_x = (v'_y + v'_z)/2
        Round each value of v to 7r bits
    v ⟵ SolveDGG(G, v)
    v ⟵ KwekMelhorn(v, 2^r)
    return v
```

T is the number of steps needed to guarantee the convergence of the algorithm. They proved that for $r \geq 6$, T can be equal to $2\ln\left(2^{5r+1}\right)2^r$. The SolveDGG$(G, v)$ algorithm is the algorithm solving the SSG G where every random vertex a has been replaced by a sink of value v_a presented in [AHMS08]. This

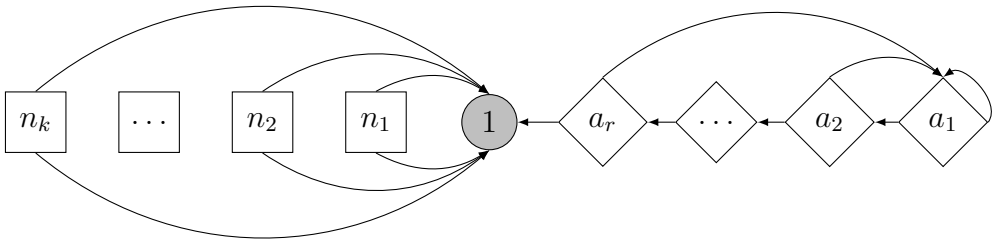


Figure 2.4: The extremal game of [IJM12] with r random nodes and k MIN vertices.

algorithm has complexity $O\left(r\log(r)+n\right)$. The KwekMelhorn algorithm presented by Kwek and Melhorn in [KM03] takes as input (v,q) and return v' with $v'_i = a/b$ the smallest fraction greater or equal than v_i and such that $b \leq q$. The call to KwekMelhorn requires time $O\left(r^2\right)$.

The main point of the proof that yields the bound of $T = 2\ln\left(2^{5r+1}\right)2^r$ is the notion of extremal games. A t-extremal game is the game that maximises the difference between its optimal value vector and the value vector computed at the step t of the algorithm. They exhibit a family of extremal games and compute the difference between the optimal value vector of this extremal game and the vector computed at step t. This allows them to conclude on the number of steps needed to be close enough to the optimal value vector. The extremal game is presented in Figure 2.4

We compare this algorithm to a family of strategy improvement algorithm in Section 4.3.2.

2.4 . Quadratic Programming Formulation

Another method to solve Simple Stochastic Games is quadratic programming. In [Con90], Condon presents the following quadratic program with linear constraints whose unique solution is the optimal value vector. This program is for 2-SSG of degree 2 with two sinks 0 and 1, but can easily be adapted for more general SSGs. For x a non sink vertex, we denote its two children by $x^{(0)}$ and $x^{(1)}$.

Minimise $F(v) = \sum_{x\in V_{\text{MAX}}\cup V_{\text{MIN}}} \left(v(x) - v(x^{(0)})(v(x) - v(x^{(1)})\right)$

under the following constraints:

$$
\begin{aligned}
v(x) &\geq v(x^{(i)}) && \text{for } x \in V_{\text{MAX}} \text{ and } i \in \{0,1\}\\
v(x) &\leq v(x^{(i)}) && \text{for } x \in V_{\text{MIN}} \text{ and } i \in \{0,1\}\\
v(x) &= \frac{1}{2}\left(v(x^{(0)}) + v(x^{(1)})\right) && \text{for } x \in V_R\\
v(0) &= 0\\
v(1) &= 1
\end{aligned}
$$

We notice that for the solution of this quadratic program to be the optimal value vector, the game needs to be stopping. Otherwise, nodes that do not reach a sink could have arbitrary value. In [KRSW22], a way to get rid of the stopping condition is provided. Their solution can create a quadratic program of exponential size compared to the original game. However, they claim that in practice it is generally better than the quadratic increase in size when creating the ϵ-transformation of the game.

For a stopping game, the objective function is always non-negative and is equal to 0 if and only if v is the optimal value vector of G. This can be proven using Lemma 1.28.

3 - The Strategy Improvement Method

Contents

In addition to value iteration and quadratic programming, explained in Chapter 2, the third and most popular method used to solve simple stochastic games is strategy improvement algorithm (SIA). It consists in considering a positional MAX strategy σ and a best response $\tau(\sigma)$, then by making some local improvement on the MAX strategy, obtaining a strategy σ' with better vector value. Repeating this step, we obtain a sequence of improving strategies. That method stems from Howard [How60] and an algorithm by Hoffman and Karp used to solve Markov Decision Process [HK66]. We present their algorithm in more details in Chapter 4.

For SSGs with MAX vertices of out-degree 2, the Hoffmann-Karp's algorithm is a deterministic SIA which makes at worst $O\left(2^n/n\right)$ iterations (see [TVK11]). This is worse than the best known deterministic algorithm, the Fibonacci Seesaw algorithm [SW01] that we present in Chapter 6 that runs in at most $O\left(1.61^n\right)$ iterations and from which we give a strategy improvement variant in Chapter 5. The best known randomised algorithm is a SIA described by Ludwig in [Lud95], which runs in $2^{O(\sqrt{n})}$.

Gimbert and Horn give an SIA in [GH08], running in $O^*\left(r!\right)$ iterations, namely a superpolynomial dependency in r only (O^* omits polynomial factors in r and n).

Auger, Coucheney and Strozecki propose in [ACS14] polynomial time algorithm for solving some families of SSG which are almost acyclic.

In [ABdMS21] we presented a meta-algorithm encompassing all strategy improvement algorithms, in order to study them all at the same time. It turns out that all SIA run in $O^*\left(2^r\right)$ iterations on 2-SSGs.

3.1 . The Generic Strategy Improvement Algorithm

In all this chapter, we consider positional strategies unless stated otherwise.

One of the goal of designing a meta-algorithm to describe strategy improvement algorithms is to prove the correctness of all previously studied SIAs for non-stopping games. By relaxing the stopping constraints, we need to consider a tighter definition of what is a better strategy. We recall from Section 2.2 that any SSG can be transformed in an equivalent stopping SSG by adding a quadratic number of random vertices. Since we provide bounds parametrised by r, we want to avoid this size blow-up. In all our proofs, we must avoid creating cycles of value 0. To do so, we introduce a new order on the strategies.

Definition 3.1. *Let σ and σ' be two* MAX *strategies, then $\sigma' \underset{G}{\succ} \sigma$ if $\sigma' \underset{G}{\geq} \sigma$ and for every* MAX *vertex x, if $v^G_{\sigma'}(x) = v^G_\sigma(x)$, then $\sigma'(x) = \sigma(x)$.*

This order will be use later on in order to consider strategies that are unchanged on some vertices when not strictly increasing the value of the vertex. When clear from the context, the game G will be omitted in the notation.

3.1.1 . Game Transformation

As stated before, strategy improvement algorithm works by finding some local improvements on a strategy that will ensure that we obtain a globally better strategy. In order to describe local improvement, we present a transformation of the game where some arcs of the game are rerouted to new sinks with appropriate values.

Definition 3.2. *Let G be an SSG, A be a subset of the arcs of G and f be a function from A to the set of rational numbers. Let $G[A, f]$ be the SSG obtained from a copy of G with the following modifications:*

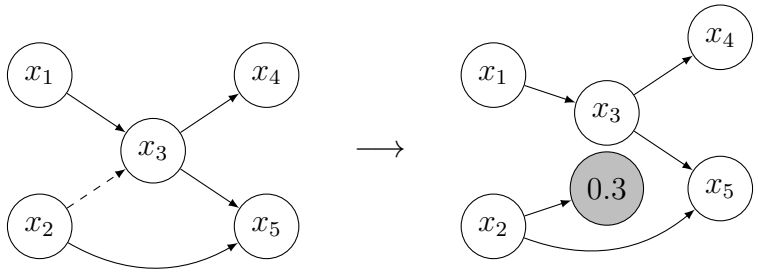


Figure 3.1: Transformation of the graph G in $G[\{(x_2, x_3)\}, f]$ where $f((x_2, x_3)) = 0.3$

each arc $e = (x, y) \in A$ is removed and replaced in $G[A, f]$ by $e' = (x, s_e)$ where s_e is a new sink vertex with value $f(e)$. These new sinks of $G[A, f]$ are called A-sinks, and A is called the set of fixed arcs.

An instance of this transformation is provided in Figure 3.1. Note that in the previous definition, the end vertex y of an arc $(x, y) \in A$ is not removed from the game. Its incoming arcs which are in A are simply redirected to sinks.

The function f is usually given by the values associated to a MAX strategy: we denote by $G[A, \sigma]$ the game $G[A, f]$, where f is defined on every arc $e = (x, y)$ of A by $f(e) = v_\sigma(y)$. Comparing G and $G[A, \sigma]$, the only differences are that arcs of A have their endpoints changed to new sinks. Therefore, a strategy defined in G can be interpreted as a strategy of $G[A, \sigma]$ and vice versa, and we identify strategies in G and $G[A, \sigma]$. Moreover, when we compare the values of a strategy in both games (as in Lemma 3.4 below), it makes sense to compare only the values on vertices in G and not on A-sinks (and anyway values of A-sinks are fixed).

As we will see later in Lemma 3.7, this transformation is equivalent to considering the transformed game where once we cross arcs from A, player MAX has to play according to the strategy σ.

We are now interested in the null sets of G and $G[A, \sigma]$ under σ. We recall that the null sets of a game defined in Definition 1.13 under a MAX strategy σ is the set of vertices x such that $v_\sigma(x) = 0$ and is denoted by K_σ.

Lemma 3.3. *For an SSG G, a subset of arcs A, and a MAX strategy σ, $K_\sigma^G = K_\sigma^{G[A,\sigma]}$.*

Proof. For $G = \left(V^G, E^G\right)$ an SSG, fix a MIN strategy τ and define $R^G_{\sigma,\tau}(x)$ as the set of vertices that can be reached from x in G, following only arcs corresponding to σ and τ after MAX and MIN vertices, and any arc out of random vertices. We repeatedly use the easy fact that the three following assertions are equivalent:

(i) $v^G_{\sigma,\tau}(x) = 0$;

(ii) $v^G_{\sigma,\tau}(y) = 0$ for all $y \in R^G_{\sigma,\tau}(x)$;

(iii) $\mathrm{Val}^G(s) = 0$ for all $s \in V_S^G \cap R^G_{\sigma,\tau}(x)$.

The same equivalence is true in $G[A, \sigma] = \left(V^{G[A,\sigma]}, E^{G[A,\sigma]}\right)$, where we define $R^{G[A,\sigma]}_{\sigma,\tau}$ likewise. Denote by $R^G_A(x)$ vertices of $R^G_{\sigma,\tau}(x)$ that are endpoints of arcs in A, and let $S_A(x)$ be the corresponding A-sinks in $G[A, \sigma]$.

Suppose that $v^G_{\sigma,\tau}(x) = 0$ and consider a sink s in $V_S^{G[A,\sigma]} \cap R^{G[A,\sigma]}_{\sigma,\tau}(x)$: either it belongs to V_S^G hence also to $R^G_{\sigma,\tau}(x)$ and satisfies $\mathrm{Val}^G(s) = 0$ by (iii), or it belongs to $S_A(x)$ and then by definition:

$$\mathrm{Val}^{G[A,\sigma]}(s) = v^G_\sigma(s) \leq v^G_{\sigma,\tau}(s) = 0.$$

Thus, by (iii) once again we have $v^{G[A,\sigma]}_{\sigma,\tau}(x) = 0$.

Conversely, suppose that $v^{G[A,\sigma]}_{\sigma,\tau}(x) = 0$ and let $s \in V_S^G \cap R^G_{\sigma,\tau}(x)$. Then, either $s \in R^{G[A,\sigma]}_{\sigma,\tau}$, hence by (iii)

$$\mathrm{Val}^G(s) = \mathrm{Val}^{G[A,\sigma]}(s) = 0,$$

or there is a $y \in R^G_A(x)$ such that $s \in R^G_{\sigma,\tau}(y)$. In this case we have $v^G_{\sigma,\tau}(y) = 0$ by (ii), hence $\mathrm{Val}^G(s) = 0$ by (iii) applied to y, and we see that $v^G_{\sigma,\tau}(x) = 0$.

Since we have $v^G_{\sigma,\tau}(x) = 0$ if and only if $v^{G[A,\sigma]}_{\sigma,\tau}(x) = 0$, regardless of τ, the result follows. □

We can now use the optimality condition presented in Lemma 1.22 to compare the value of G and $G[A,\sigma]$ under σ.

Lemma 3.4. *For an SSG G, a subset of arcs A, and a* MAX *strategy σ, $v^G_\sigma = v^{G[A,\sigma]}_\sigma$.*

Proof. This is a direct consequence of Lemma 1.22 and Lemma 3.3, since the vector v^G_σ satisfies the best-response conditions in $G[A,\sigma]$ and vice versa. □

This lemma implies that if player MAX plays following strategy σ in $G[A,\sigma]$, they will obtain the same expected gain than if they play as σ in G. Thus, with this game transformation, **the concept of improving locally is redefined as improving in** $G[A,\sigma]$ for some set of arcs A.

3.1.2 . The Algorithm

Algorithm 3 is a classical strategy improvement algorithm with two twists: the improvement is for *the stricter order* $\succ$ and it is guaranteed in *the transformed game* rather than in the original game. We call Algorithm 3 the Generic Strategy Improvement Algorithm, or GSIA. We recall that we denote by $\tau(\sigma)$ a positional best response to σ.

Algorithm 3: GSIA

Data: G an SSG
Result: (σ, τ) a pair of optimal strategies

```
begin
    select an initial MAX strategy σ
    while (σ, τ(σ)) are not optimal strategies of G do
        choose a subset A of arcs of G
        find σ' such that σ' ≻_{G[A,σ]} σ.
        σ ⟵ σ'
    return (σ, τ(σ))
```

Algorithm 3 is a generic algorithm (or meta-algorithm) because neither the selection of an initial strategy σ at line 2, nor the way of choosing A at line 4, nor the way of finding σ' at line 5, are specified. A choice of implementation for these three parts is an **instance** of GSIA, that is a concrete strategy improvement algorithm. To obtain an instance of GSIA with a good complexity, the choice of the subset A must both ensure that a better strategy in $G[A,\sigma]$ can be found efficiently (e.g. in polynomial time) and that there is a small number of iterations.

Note that if $\sigma' \underset{G[A,\sigma]}{>} \sigma$ is found, it is easy to find σ'' with $\sigma'' \underset{G[A,\sigma]}{\succ} \sigma$: define σ'' as equal to σ', except for MAX vertices x such that $v^G_{\sigma'}(x) = v^G_\sigma(x)$ and $\sigma'(x) \neq \sigma(x)$ where $\sigma''(x)$ is defined as $\sigma(x)$. By Lemma 1.22, σ'' has a different vector value than σ' only if they have a different null set, thus if a switched x is in $K_{\sigma''}$ and not in $K_{\sigma'}$. However, this would imply that x is in K_σ, thus by definition x is in $K_{\sigma'}$.

When we prove some property of GSIA in this chapter, it means that the property is **true for all instances of GSIA**, that is, regardless of the selection of the initial strategy, the set A and the method for selecting σ'.

In order to prove the correctness of GSIA, we need to prove two points:

1. If σ is not optimal in G, then σ is not optimal in $G[A,\sigma]$.
2. If $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$ then $\sigma' \underset{G}{\geq} \sigma$.

The first point is proved in the following lemma, while the second one is harder to obtain and is the subject of the next two subsections.

Lemma 3.5. *For an SSG G and a subset of arcs A, a* MAX *strategy σ is optimal in G if and only if it is optimal in $G[A,\sigma]$.*

Proof. Except on A-sinks, the value vectors of σ in G and $G[A,\sigma]$ are equal by Lemma 3.4. Furthermore, by Lemma 3.3, $K^G_\sigma = K^{G[A,\sigma]}_\sigma$; hence σ satisfies the optimality conditions of Lemma 1.28 in G if and only if it satisfies them in $G[A,\sigma]$. □

3.2 . Proof of Correctness of GSIA

3.2.1 . Concatenation of Strategies

As a tool for proving the correctness of Algorithm 3, we introduce the notion of concatenation of strategies, which produces non-positional strategies even if both concatenated strategies are positional. The idea of using a sequence of concatenated strategies to interpolate between two strategies has been introduced in [GH09].

Definition 3.6. *For two* MAX *strategies σ, σ' and a subset of arcs A, we call $\sigma'|_A\sigma$ the non-positional strategy that plays like σ' until an arc of A is crossed, and then plays like σ until the end of the game. We let $\sigma'|^0_A\sigma = \sigma$ and for all $i \geq 0$, $\sigma'|^{i+1}_A\sigma = \sigma'|_A(\sigma'|^i_A\sigma)$.*

When A is clear from the context, we omit it and write $\sigma'|^i\sigma$. Strategy $\sigma'|^i_A\sigma$ is the strategy that plays like σ' until i arcs from A have been crossed, and then plays like σ. We can relate the strategy $\sigma'|_A\sigma$ to a positional strategy in $G[A,\sigma]$, as shown in the next lemma.

Lemma 3.7. *For two* MAX *strategies σ, σ' and a subset of arcs A, we have:* $v^G_{\sigma'|_A\sigma} = v^{G[A,\sigma]}_{\sigma'}$

Proof. In G, after crossing an arc from A, by definition of $\sigma'|_A\sigma$, MAX plays according to σ. The game being memoryless, from this point, the best response for player MIN is to play like $\tau(\sigma) \in BR(\sigma)$. Thus, there is a best response to $\sigma'|\sigma$ of the form $\tau'|\tau(\sigma)$ with τ' a MIN strategy not necessarily positional. Let us consider a play following $(\sigma'|\sigma, \tau|\tau(\sigma))$ with τ any MIN strategy. If the play does not cross an arc of A, then there is no difference between this play and a play following (σ', τ) in $G[A, \sigma]$. If an arc of A is used, then by Lemma 3.4 there is no difference between stopping with the value of $G[A, \sigma]$ or continuing in G while following (σ, τ). Thus, we have: $v^G_{\sigma'|\sigma,\tau|\tau(\sigma)} = v^{G[A,\sigma]}_{\sigma',\tau}$.

Thus, if τ' is a best response to σ' in $G[A, \sigma]$, then $\tau'|\tau(\sigma)$ is a best response to $\sigma'|\sigma$ in G. This implies that $v^G_{\sigma'|\sigma} = v^{G[A,\sigma]}_{\sigma'}$. □

We now prove the fact that increasing the values of sinks can only increase the value of the game (a similar lemma is proved in [ACS14]).

Lemma 3.8. *Let G and G' be two identical SSGs except the values of theirs sinks $s \in V_S$, denoted respectively by $Val(s)$ and $Val'(s)$. If for every $s \in V_S$, $Val'(s) \geq Val(s)$, then for every* MAX *strategy σ we have $v^{G'}_\sigma \geq v^G_\sigma$.*

Proof. For $s \in V_S$, let $\mathbb{P}^x_{\sigma,\tau}(\to s)$ be the probability that the play ends in sink s while starting from vertex x, following strategies (σ, τ). For any vertex x we have:

$$v^{G'}_{\sigma,\tau}(x) = \sum_{s\in V'_S} \mathbb{P}^x_{\sigma,\tau}(\to s)\mathrm{Val}'(s) \geq \sum_{s\in V'_S} \mathbb{P}^x_{\sigma,\tau}(\to s)\mathrm{Val}(s) = v^G_{\sigma,\tau}(x)$$

This is true for any MIN strategy τ, thus $v^{G'}_\sigma \geq v^G_\sigma$. □

3.2.2 . Absorbing Set

The following proposition is the core idea of GSIA: **a strategy which improves on σ in the transformed game also improves on σ in the original game**. The proof relies on a precise analysis of the set of vertices which cannot reach a sink, to deal with the fact that the game is not stopping. We prove that, if $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$ the limit of $v^G_{\sigma'|^i\sigma}$ is $v^G_{\sigma'}$ and the two previous lemmas imply $\sigma'|^i\sigma \geq \sigma'|^{i-1}\sigma > \sigma$, which yields the following proposition.

Proposition 3.9. *Let G be an SSG, A a subset of arcs of G and σ, σ' two* MAX *strategies. If $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$ then $\sigma' \underset{G}{\geq} \sigma$.*

In order to avoid requiring the game to be stopping, it is necessary to pay particular attention to the set of vertices where the play can loop infinitely and yield value zero, which is a subset of the set of vertices of value 0. We now prove that a step of GSIA can only reduce this set, which is then used to prove Proposition 3.9.

Definition 3.10. *For an SSG G and two strategies (σ, τ), an absorbing set Z is a subset of $V \setminus V_S$ such that starting from any vertex of Z and playing according to (σ, τ), the vertices of $V \setminus Z$ are unreachable.*

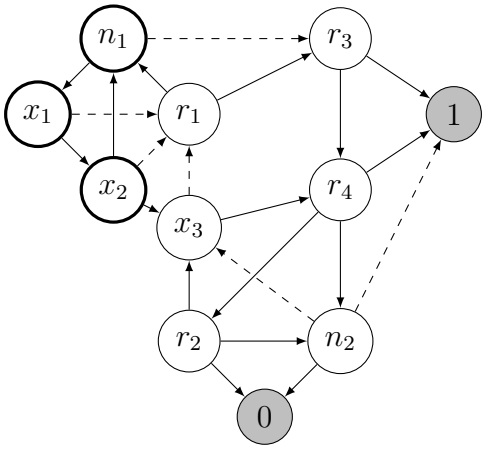


Figure 3.2: Example of an SSG where the x, n and r vertices are respectively from V_{MAX}, V_{MIN} and V_R. The pair of strategy (σ, τ) is displayed as plain arrows. Here $Z(\sigma, \tau) = Z(\sigma) = \{n_1,\ x_1,\ x_2\}$.

For σ and τ two strategies, $Z(\sigma, \tau)$ is the set of all vertices in some absorbing set under (σ, τ). Hence, $Z(\sigma, \tau)$ is also an absorbing set. By definition, a play remains stuck in an absorbing set and can never reach a sink, hence all vertices of an absorbing set have *value zero* under (σ, τ). The next lemma proves the existence of the inclusion-wise maximum over τ of $Z(\sigma, \tau)$ that we denote by $Z(\sigma)$. An example is given in Figure 3.2.

Lemma 3.11. *For every* MAX *strategy* σ*, there is* $\tau \in BR(\sigma)$ *such that for every* MIN *strategy* τ'*, we have* $Z(\sigma, \tau') \subseteq Z(\sigma, \tau)$.

Proof. For τ in $BR(\sigma)$ and τ' such that $Z(\sigma, \tau') \nsubseteq Z(\sigma, \tau)$, then we define $\tilde{\tau}$ as $\tilde{\tau}(x) = \tau'(x)$ for x in $Z(\sigma, \tau')$ and $\tilde{\tau}(x) = \tau(x)$ otherwise. We now prove that $\tilde{\tau} \in BR(\sigma)$ and $Z(\sigma, \tilde{\tau}) \supseteq Z(\sigma, \tau) \cup Z(\sigma, \tau')$.

Since τ is a best response to σ, we have $v_{\sigma,\tau}(x) \leq v_{\sigma,\tau'}(x)$. Moreover, for $x \in Z(\sigma, \tau')$, $v_{\sigma,\tau'}(x) = 0$ thus $v_{\sigma,\tau}(x) = 0$. From this, we deduce that the two systems of linear equations given by Lemma 1.22, characterising respectively vectors $v_{\sigma,\tau}$ and $v_{\sigma,\tilde{\tau}}$, are exactly the same: the only vertices where $\tilde{\tau}(x)$ and $\tau(x)$ differ satisfy $v_{\sigma,\tau}(\tau(x)) = v_{\sigma,\tau}(\tilde{\tau}(x)) = 0$. Hence, we have $v_{\sigma,\tau} = v_{\sigma,\tilde{\tau}}$ and $\tilde{\tau} \in BR(\sigma)$.

For any play under strategies $(\sigma, \tilde{\tau})$ starting in $x \in Z(\sigma, \tau')$, the MIN vertices of the play are all in $Z(\sigma, \tau')$ because $\tilde{\tau}$ plays as τ' on these vertices. Thus, we have $Z(\sigma, \tau') \subseteq Z(\sigma, \tilde{\tau})$. For a play starting in $x \in Z(\sigma, \tau)$, either the play reaches a vertex of $Z(\sigma, \tau')$ and then stays in $Z(\sigma, \tau')$ or it plays like τ and stays in $Z(\sigma, \tau)$. Hence, we have $Z(\sigma, \tau) \subseteq Z(\sigma, \tilde{\tau})$. □

From this we deduce the following result on the improvement step for GSIA (where absorbing sets are understood in G):

Proposition 3.12. *Let* G *be an SSG,* A *a set of arcs of* G*,* σ *and* σ' *two* MAX *strategies such that* $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$*, then* $Z(\sigma') \subseteq Z(\sigma)$.

Proof. From Lemma 3.11 there is $\tau \in BR(\sigma)$ such that $Z(\sigma, \tau) = Z(\sigma)$ and $\tau' \in BR(\sigma')$ such that $Z(\sigma', \tau') = Z(\sigma')$. We write $Z = Z(\sigma')$.

Suppose that $Z(\sigma')$ is not a subset of $Z(\sigma)$. Let X be the set of MAX vertices x in $Z(\sigma') \setminus Z(\sigma)$ such that $\sigma(x) \neq \sigma'(x)$; it is non-empty otherwise $Z(\sigma')$ would be an absorbing set for (σ, τ'). If x is in X, since $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$, we have

$$v_{\sigma'}^{G[A,\sigma]}(x) > v_{\sigma}^{G[A,\sigma]}(x) \geq 0$$

Thus, a sink is reached in $G[A, \sigma]$ starting from x under the strategies (σ', τ'). Since Z is an absorbing set in G under the same strategies, it implies that all the accessible sinks in $G[A, \sigma]$ are A-sinks. Hence, there is at least one arc $e = (y, z) \in A$ with both ends in Z and such that $v_\sigma(z) > 0$. We define the vertex s of Z as:

$$s = \arg\max_{z \in Z} \{v_\sigma(z) \mid \exists y \in V, (y, z) \in A\}$$

and we let $v = v_\sigma(s)$. The value of each vertex in Z is bounded by v.The value of s in G under strategies (σ, τ) is bounded by the value of the vertices leaving Z, as we have explained for x. Such vertices exist since Z is not a subset of $Z(\sigma)$. We now want to show that those vertices all have value strictly less than v, thus proving a contradiction.

First, since Z is an absorbing set for (σ', τ'), all arcs leaving a random vertex in $Z(\sigma')$ remain in $Z(\sigma')$ in G; this is not dependent on the strategies considered.

Let $E_X \subseteq X$ the set of MAX vertices x of X such that $\sigma(x) \notin Z$ and let $E_N \subseteq Z \cap V_{\text{MIN}}$ the set of MIN vertices x of Z such that $\tau(x) \notin Z$.

On the one hand, for a MIN vertex $x \in E_N$:

$$\begin{aligned}
v_\sigma^G(\tau(x)) &\leq v_\sigma^G(\tau'(x)) && \text{Since } \tau = \tau(\sigma) \\
v_\sigma^G(\tau'(x)) &= v_\sigma^{G[A,\sigma]}(\tau'(x)) && \\
v_\sigma^{G[A,\sigma]}(\tau'(x)) &\leq v_{\sigma'}^{G[A,\sigma]}(\tau'(x)) && \text{Since } \sigma' \underset{G[A,\sigma]}{\succ} \sigma \\
v_{\sigma'}^{G[A,\sigma]}(\tau'(x)) &\leq v && \text{Since } \tau'(x) \in Z
\end{aligned}$$

Thus, $v_\sigma^G(\tau(x)) \leq v$. In case of equality, we have $v = v_\sigma^G(\tau'(x)) = v_\sigma^G(\tau(x))$; hence we can replace τ by $\bar{\tau}(x)$,which is identical to τ except that $\bar{\tau}(x) = \tau'(x)$. We have $v_{\sigma,\tau} = v_{\sigma,\bar{\tau}}$ and $Z(\sigma, \tau) = Z(\sigma, \bar{\tau})$. Indeed, according to Lemma 1.22 the only situation that could occur would be to violate the condition (v) by creating an absorbing set. However, this would contradict the definition of τ. Thus, we can suppose that for any x in E_N, $v_\sigma(\tau(x)) < v$.

On the other hand, since $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$ we know that for any x in E_X:

$$v_\sigma^G(\sigma(x)) < v_{\sigma'}^{G[A,\sigma]}(x) \leq v$$

Now, for any vertex x of $E = E_X \cup E_N$, let p_x be the probability of x being the first vertex of E reached starting from s following strategies (σ, τ). By conditional expectation:

$$v_\sigma(s) = \sum_{x \in E_X} p_x v_\sigma(\sigma(x)) + \sum_{x \in E_N} p_x v_\sigma(\tau(x))$$

Thus, $v_\sigma(s) < v$ which contradicts the definition of v, and proves that $Z(\sigma') \subseteq Z(\sigma)$. □

We now prove Proposition 3.9.

Proof. We introduce a sequence of non-positional strategies $(\sigma_i)_{i\geq 0}$ defined by $\sigma_i = \sigma'|^i\sigma$ for $i \geq 1$. By hypothesis $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$, and by Lemma 3.7 $v^G_{\sigma'|_A\sigma} = v^{G[A,\sigma]}_{\sigma'}$, then we have

$$v^G_{\sigma_1} = v^G_{\sigma'|\sigma} = v^{G[A,\sigma]}_{\sigma'} > v^{G[A,\sigma]}_{\sigma} = v^G_\sigma.$$

Hence, by definition, sinks of $G[A,\sigma_1]$ will have at least the values of the corresponding sinks in $G[A,\sigma]$. Applying Lemma 3.8, we obtain that $v^{G[A,\sigma_1]}_{\sigma'} \geq v^{G[A,\sigma]}_{\sigma'}$, which can also be written as $v^G_{\sigma_2} \geq v^G_{\sigma_1}$. More generally, we have:

$$\forall i \geq 1, v^G_{\sigma_{i+1}} \geq v^G_{\sigma_i} \geq v^G_{\sigma_1} > v^G_\sigma.$$

We now prove that $v^G_{\sigma'} \geq v^G_{\sigma_1}$ to conclude the proof.

From now on, we only consider the game G. Fix a vertex x and a MIN strategy $\tau \in BR(\sigma')$ such that $Z(\sigma') = Z(\sigma',\tau)$. From Proposition 3.12 we know that, $Z(\sigma') \subseteq Z(\sigma)$. It implies that for every $z \in Z(\sigma')$, $v^G_\sigma(z) = v^G_{\sigma'}(z) = 0$ which implies that $v^{G[A,\sigma]}_{\sigma'}(z) = 0$. Thus, $\sigma'(z) = \sigma(z)$. It implies that $Z(\sigma') \subseteq Z(\sigma,\tau)$.

We now only consider G' the game G where we replace every vertex in $Z(\sigma',\tau)$ by a sink of value 0. Lemma 3.4 directly implies that $v^G_\sigma = v^{G'}_\sigma$ and $v^G_{\sigma'} = v^{G'}_{\sigma'}$. Moreover, when playing following σ_i when a vertex of $Z(\sigma')$ is reached, for all possible history, the play will stay in the absorbing set. Thus, $v^G_{\sigma_i} = v^{G'}_{\sigma_i}$.

Recall that $\mathbb{P}^x_{\sigma',\tau}(\to s)$ is the probability to reach a sink s in G' while starting in x and following (σ',τ). Let $T^{\sigma',\tau}$ be a random variable defined as the time at which a sink is reached. Note that $T^{\sigma',\tau}$ may be equal to $+\infty$.

For every $i \geq 1$, we use Bayes rule to express the value of $v_{\sigma',\tau}(x)$ while conditioning on finishing the game before i steps.

$$\begin{aligned} v_{\sigma',\tau}(x) =& \mathbb{P}(T^{\sigma',\tau} < i) \sum_{s\in V_S} \mathbb{P}^x_{\sigma',\tau}(\to s \mid T^{\sigma',\tau} < i)\mathrm{Val}(s) \\ &+ \mathbb{P}(i \leq T^{\sigma',\tau} < +\infty) \sum_{s\in V_S} \mathbb{P}^x_{\sigma',\tau}(\to s \mid +\infty > T^{\sigma',\tau} \geq i)\mathrm{Val}(s) \end{aligned}$$

If $T^{\sigma_i,\tau} < i$, only i arcs have been crossed, thus at most i arcs from A have been crossed when the sink is reached. Hence, σ_i acts like σ' during the whole play, which yields:

$$\begin{aligned} v_{\sigma',\tau}(x) =& \mathbb{P}(T^{\sigma_i,\tau} < i) \sum_{s\in V_S} \mathbb{P}^x_{\sigma_i,\tau}(\to s \mid T^{\sigma_i,\tau} < i)\mathrm{Val}(s) \\ &+ \mathbb{P}(i \leq T^{\sigma',\tau} < +\infty) \sum_{s\in V_S} \mathbb{P}^x_{\sigma',\tau}(\to s \mid +\infty > T^{\sigma',\tau} \geq i)\mathrm{Val}(s) \end{aligned}$$

We use Bayes rule in the same way for $v_{\sigma_i,\tau}(x)$

$$\begin{aligned} v_{\sigma_i,\tau}(x) =& \mathbb{P}(T^{\sigma_i,\tau} < i) \sum_{s \in V_S} \mathbb{P}^x_{\sigma_i,\tau}(\to s \mid T^{\sigma_i,\tau} < i)\mathrm{Val}(s) \\ &+ \mathbb{P}(i \leq T^{\sigma_i,\tau} < +\infty) \sum_{s \in V_S} \mathbb{P}^x_{\sigma_i,\tau}(\to s \mid T^{\sigma_i,\tau} \geq i)\mathrm{Val}(s) \end{aligned}$$

Since every absorbing vertex in G associated with σ' has been turned into a sink, in G', $\mathbb{P}(T^{\sigma',\tau} < i) = \mathbb{P}(T^{\sigma_i,\tau} < i)$ converges to 1 when i grows. Hence, both $\mathbb{P}(i \leq T^{\sigma',\tau} < +\infty)$ and $\mathbb{P}(i \leq T^{\sigma_i,\tau} < +\infty)$ go to 0 and

$$\lim_{i \to +\infty} |v_{\sigma',\tau}(x) - v_{\sigma_i,\tau}(x)| = 0.$$

Hence, if there was x such that $v_{\sigma'}(x) < v_{\sigma_1}(x)$, we denote $\epsilon = v_{\sigma_1}(x) - v_{\sigma'}(x)$. For some rank I for all $i \geq I$ we have $|v_{\sigma',\tau} - v_{\sigma_i,\tau}| < \epsilon/2$. Which implies $v_{\sigma_i,\tau}(x) < v_{\sigma_1}(x)$. We recall that $v_{\sigma_1}(x) \leq v_{\sigma_i}(x)$. This means that $v_{\sigma_i,\tau} < v_{\sigma_i}(x)$, which contradicts the notion of optimal response against σ_i. Therefore, we have shown that $\sigma' \underset{G}{\geq} \sigma_1 \underset{G}{>} \sigma$. □

As a consequence of all previous lemmas, we obtain the correctness of GSIA.

Theorem 3.13. *GSIA terminates and returns a pair of optimal strategies.*

Proof. We denote by σ_i the MAX strategy σ at the end of the i^{th} loop in Algorithm 3. By induction, we prove that the sequence σ_i is of increasing value. Indeed, Line 5 of Algorithm 3 guarantees that $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$, thus Proposition 3.9 implies that $\sigma' \underset{G}{>} \sigma$, that is $\sigma_{i+1} > \sigma_i$.

The strategies produced by the algorithm are positional, hence there is only a *finite number of them*. Since the sequence is strictly increasing, it stops at some point. The algorithm only stops when the while condition of Line 3 is not satisfied, hence when $(\sigma, \tau(\sigma))$ is optimal in G. □

3.3 . Switch and Switch set

We now use GSIA in order to present an important concept in strategy improvement method: the switch. This concept stems from the study of Markov Decision Process by Howard [How60] and an extension to stochastic games by Hoffman and Karp [HK66]. It has been adapted for Simple Stochastic Games by Condon in [Con90] and further studied by Tripathi, Valkanova and Kumar in [TVK11].

3.3.1 . Definition

Definition 3.14 (Switch Set)**.** *Let $G = (V, E)$ be an SSG and σ a* MAX *strategy. The switch set of σ, written S_σ, is the set of vertices x such that there is y with $(x, y) \in E$ and $v_\sigma(x) < v_\sigma(y)$*

Definition 3.15 (Improvement set)**.** *Let σ be a* MAX *strategy. For all $x \in V_{\text{MAX}}$, the improvement set of x under σ, denoted by $IS_\sigma(x)$, is the set of neighbours y of x such that $v_\sigma(y) > v_\sigma(x)$ and the best improvement option is defined as $bio_\sigma(x) = \underset{y \in IS_{\sigma(x)}}{\arg\max}\{v_\sigma(y)\}$.*

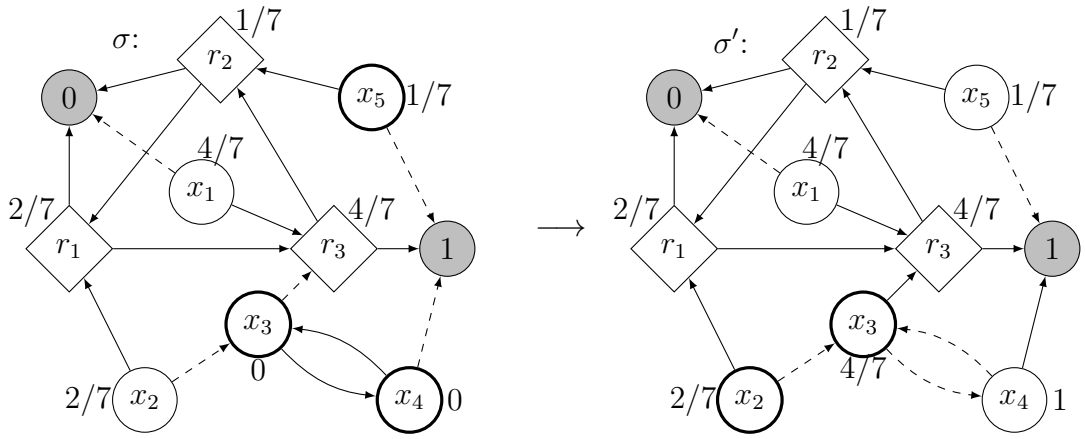


Figure 3.3: The strategies of MAX are represented by plain arcs, and the probability distribution on the random vertices is the uniform distribution. The switch set S_σ of σ is $\{x_3, x_4, x_5\}$. The strategy σ' is a σ-switch.

In other words, the switch set of a MAX strategy is the set of MAX vertices that do not satisfy the local optimality condition presented in Lemma 1.28. It is clear that if $x \notin S_\sigma$ then, $IS_\sigma(x) = \emptyset$. This notion directly gives the concept of σ-switch.

Definition 3.16 (Switch)**.** *Let σ be a* MAX *strategy with a non-empty switch set S_σ. A* MAX *strategy σ' is said to be a σ-switch or a switch of σ, if $\sigma' \neq \sigma$, and for all $x \in V_{\text{MAX}}$ such that $\sigma'(x) \neq \sigma(x)$, $\sigma'(x) \in IS_\sigma(x)$.*

A σ-switch is a strategy, where the strategy on vertices that satisfy local optimality for σ has been kept, and it had been changed on some vertices that did not satisfy local optimality. Informally, as the name implies, it is a strategy where we "switch" the strategy on some vertices in order to achieve an immediate better value. A representation of a switch is given in Figure 3.3.

Conversely, there is also the opposite concept of locally worsening the strategy. This is a σ-anti-switch.

Definition 3.17. *Let σ be a* MAX *strategy. A* MAX *strategy σ' is said to be a σ-anti-switch or an anti-switch of σ, if $\sigma' \neq \sigma$, and for all $x \in V_{\text{MAX}}$ such that $\sigma'(x) \neq \sigma(x)$, $\sigma'(x) \notin IS_\sigma(x)$.*

A common tool to solve SSGs is the fact that **a switch increases the value of a strategy, while an anti-switch decreases it**. This was first proven by Condon for stopping games in [Con90]. A more technical proof for general SSG was given by Chatterjee, de Alfaro and Henzinger in [CdH13]. We can note that even the proof of Condon in the case of stopping game is technical whereas within the framework of GSIA and transformed game, it is extremely simple to prove as we showed in [ABdMS21].

Lemma 3.18. *If σ_S is a switch of σ, then $\sigma_S > \sigma$. If σ_S is an anti-switch of σ, then $\sigma_S \leq \sigma$.*

Proof. Consider $G[A, \sigma]$ the game obtained from G, where A is the set of all arcs of G. Let us consider x a vertex switched in σ', that is with $v_\sigma^G(\sigma(x)) < v_\sigma^G(\sigma'(x))$. Then, because all arcs are in

A, we have $v_\sigma^{G[A,\sigma]}(x) = v_\sigma^G(\sigma(x))$ and $v_{\sigma'}^{G[A,\sigma]}(x) = v_\sigma^G(\sigma'(x))$. Hence, $v_\sigma^{G[A,\sigma]}(x) < v_{\sigma'}^{G[A,\sigma]}(x)$ and for $v_\sigma^G(\sigma(x)) \geq v_\sigma^G(\sigma'(x))$, $\sigma(x) = \sigma'(x)$, which implies $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$. Proposition 3.9 proves $\sigma' \underset{G}{\geq} \sigma$.

The proof is the same for an anti-switch, since $\sigma \underset{G[A,\sigma]}{\succ} \sigma' \Rightarrow \sigma \underset{G}{\geq} \sigma'$ which can be proved similarly as Proposition 3.9, while keeping in mind that in the decreasing case, creating absorbing set lowers the value. □

This immediately gives the following corollary.

Corollary 3.19. *A positional* MAX *strategy σ is optimal if and only if S_σ is empty.*

Proof. If σ satisfies $S_\sigma = \emptyset$, then every positional strategy σ' is an anti-switch of σ and $\sigma' \leq \sigma$. □

3.3.2 . An Upper Bound on the Number of Iteration of GSIA

GSIA produces a sequence of strictly increasing positional MAX strategies. The number of positional MAX strategies is bounded by $|\Sigma^{\text{MAX}}| = \prod_{x \in V_{\text{MAX}}} \deg(x)$, hence the number of iterations of GSIA is bounded by this value. More specific bounds are given for some algorithms in Chapter 4, which deals with different instances of strategy improvement algorithms, as well as in Chapter 6 which presents SSG as a unique sink orientation problem.

We give a bound for q-SSG, which depends on q and r the number of random vertices. The difference of two values written as a/b and c/d, with a and b less than q^{-r} is more than q^{-2r}. Hence, if a value increases in GSIA, it increases at least by q^{-2r}. Using the classical notion of switch and anti-switch ([TVK11]), recalled previously, we can prove that all vertices which have their value increased by a step of GSIA, are increased by at least q^{-r}.

Theorem 3.20 (Global bound on GSIA)**.** *For G a q-SSG with r random vertices and n* MAX *vertices, the number of iterations of GSIA is at most nq^r.*

Proof. Let us consider σ the strategy computed at some point by GSIA and σ' the next strategy. By Proposition 3.9, $\sigma < \sigma'$. Hence, by Lemma 3.18, σ' cannot be an anti-switch of σ. Thus, there is a MAX vertex x such that $v_\sigma(\sigma(x)) < v_\sigma(\sigma'(x))$. We recall that $\sigma'(x)$ denotes the successor of x under strategy σ'.

Since $\sigma < \sigma'$, we have $v_\sigma(x) = v_\sigma(\sigma(x)) < v_\sigma(\sigma'(x)) \leq v_{\sigma'}(\sigma'(x)) = v_{\sigma'}(x)$. We now evaluate $v_\sigma(\sigma'(x)) - v_\sigma(\sigma(x))$. In the game G, under the strategies $(\sigma, \tau(\sigma))$, Theorem 2.1 implies that for some $t \leq q^r$, $v_\sigma(\sigma(x)) = p/t$ and $v_\sigma(\sigma'(x)) = p'/t$. We have $p/t < p'/t$, thus $p'/t - p/t \geq 1/t \geq 1/q^r$. Hence, the value of some MAX vertex increases by $1/q^r$ in each iteration of GSIA. Since there are n MAX vertices and their values are bounded by 1, there are at most nq^r iterations. □

The complexity of GSIA is the number of iterations given by Theorem 3.20, multiplied by the complexity of an iteration. In an iteration, there are two sources of complexity: constructing the game $G[A,\sigma]$ and finding an improving strategy σ' in $G[A,\sigma]$. To construct the game, v_σ is computed by solving a linear program of size m up to precision $p = q^r$. Let $C_1(m,p)$ be the complexity of computing v_σ, then the best bound is currently in $O(m^\omega \log(p))$ [JSWZ20], with ω the current best bound on the matrix multiplication exponent. Let $C_2(n,r,q)$ be the complexity of computing σ', the complexity of an iteration is in $O(nq^r(C_1(n+r,q^r) + C_2(n,r,q))$.

We obtain a better complexity when $C_2(n,r,q) = O(C_1(n,q^r)r/n)$, which is the case for most instances of GSIA mentioned in this thesis. The number of iterations is only rq^r if we can guarantee that a random vertex increases its value at each step. When no random vertex is improved, the cost of computing $G[A,\sigma]$ can be made smaller, which yields the following theorem.

Theorem 3.21. *Let G be a q-SSG with r random vertices and n MAX vertices. If $C_2(n,r,q) = O(C_1(n,q^r)r/n)$, then the complexity of GSIA is in $O(rq^rC_1(n,q^r))$.*

Proof. We assume that $r < n$, otherwise the theorem is trivial. Let σ' be the strategy computed by GSIA at some point, improving on the strategy σ. GSIA must compute $G[A,\sigma']$, and thus $v_{\sigma'}$ and we explain a method to do so efficiently.

We assume that the order of the values (in G) of the random vertices is the same for σ and σ'. Then, knowing this order and σ', it is easy to compute $\tau(\sigma')$ a best response to σ' in $O(r\log(r)+n)$ time [AHMS08]. Then, we can compute the values $v_{\sigma',\tau(\sigma')}$ in time $O(C_1(r,q^r))$, since it is done by solving a linear system of dimension r with precision q^r, a task which is simpler than solving a linear program. Since $C_1(r,q^r)$ is at least quadratic in r, then $C_1(r,q^r) < C_1(n,q^r)r/n$ and by hypothesis $C_2(n,r,q) = O(C_1(n,q^r)r/n)$, hence a step is of complexity at most $O(C_1(n,q^r)r/n)$. There are at most nq^r such steps, for a total complexity of $O(rq^rC_1(n,q^r))$.

We need to detect when the assumption that the values of the random vertices are the same for σ and σ' is false. If $v_{\sigma',\tau(\sigma')}$ satisfies the optimality conditions at the MIN vertices, then $\tau(\sigma')$ is a best response. Otherwise, we compute the best response by solving a linear program in time $C(n,q^r)$. In that case, the order of the random vertices has changed: there are two vertices x_1 and x_2, such that $v_\sigma(x_1) < v_\sigma(x_2)$ and $v_{\sigma'}(x_1) > v_{\sigma'}(x_2)$. Hence, $v_{\sigma'}(x_1) > v_\sigma(x_2)$, which implies that $v_{\sigma'}(x_1) - v_\sigma(x_1) > v_\sigma(x_2) - v_\sigma(x_1) > q^{-r}$.

We have proved that when the random order changes, the value of some random vertex increases by at least q^{-r}, hence there are at most rq^r such steps. The complexity of these steps is bounded by $O(rq^rC_1(n,q^r))$, which proves the theorem. □

3.4 . Max of Two Strategies

Finding a first initial strategy can significantly shorten the duration of the algorithm. This can be done by finding a strategy better than a set of given strategies. Hence, given two MAX strategies, σ and σ' it can be helpful to find a strategy $\tilde{\sigma}$ such that $\tilde{\sigma} \geq \sigma$ and $\tilde{\sigma} \geq \sigma'$. To do so, we introduce the max of two strategies.

Definition 3.22. *Let σ and σ' two positional MAX strategies. We call max strategy of σ,σ' the strategy denoted by $\sigma \wedge \sigma'$ defined by $\sigma \wedge \sigma'(x) = \sigma(x)$ if $v_\sigma(x) \geq v_{\sigma'}(x)$ and $\sigma \wedge \sigma'(x) = \sigma'(x)$ otherwise.*

Remark 3.23.

- The operator $\wedge$ is not commutative. Indeed, it is important to choose a default strategy when they both have the same value.
- The strategy $\sigma \wedge \sigma'$ can be σ or σ'.

Proposition 3.24. *The strategy $\sigma \wedge \sigma'$ has a larger value than σ and σ'.*

Proof. The first part of our proof is to notice that $\sigma \wedge \sigma'$ does not have a larger absorbing set than either σ or σ'.

For the sake of contradiction, assume that $Z = Z(\sigma \wedge \sigma') \setminus \big(Z(\sigma) \cap Z(\sigma')\big)$ is not empty. Let $x \in Z$ that maximise $\max(v_\sigma(x), v_{\sigma'}(x))$. We suppose that $\max(v_\sigma(x), v_{\sigma'}(x)) = v_\sigma(x)$. Notice that starting from x, the path following σ only encounters vertices in Z. Indeed, the max hypothesis implies that the strategy is never switched. Hence, x is in $Z(\sigma)$ which contradicts the fact that $v_\sigma(x) \geq v_{\sigma'}(x)$ and $x \notin Z(\sigma')$. Hence, $Z = \emptyset$.

We can now consider the following sequence of strategies $(\sigma \wedge \sigma')|_E^i \sigma''$ where σ'' plays as σ if the first vertex x visited satisfy $v_\sigma(x) \geq v_{\sigma'}(x)$ and σ' otherwise. The value of $(\sigma \wedge \sigma')|_E^i \sigma''$ is not decreasing, hence by a similar proof as Proposition 3.9 it implies that $\sigma \wedge \sigma' \geq \sigma$ and $\sigma \wedge \sigma' \geq \sigma'$ □

4 - Capturing Classical Algorithm with GSIA

Contents

In this chapter, we present several instances of strategy improvement algorithms, as well as lower bounds on the number of iterations of these algorithms.

4.1 . Policy Iteration Algorithm

4.1.1 . Hoffman-Karp Algorithm

In [MC90], Melekopoglou and Condon present the policy iteration method. In fact, the strategy improvement method that has been described in Chapter 3 is an extension of the policy iteration algorithm which itself comes from [How60] for solving Markov Decision Process and the Hoffman-Karp algorithm that has been developed by Hoffman and Karp in [HK66] for solving stochastic games.

A policy iteration algorithm computes a best response $\tau(\sigma)$ from a MAX strategy σ, and then selects a σ-switch. From an initial MAX strategy, it iterates these operations until finding an optimal strategy. Thus, a policy iteration algorithm is an instance of GSIA where the set of fixed arcs A is always E the set of all arcs of G.

Lemma 4.1. *Policy iteration algorithms terminate and find a pair of optimal strategies.*

Proof. The policy iteration method is a restriction of the strategy improvement method by Lemma 3.18, hence Theorem 3.13 directly proves that policy iteration algorithms terminate and find a pair of optimal strategy. □

We recall that n is the number of MAX vertices.

Lemma 4.2. *There is a policy iteration algorithm that terminates in less than n iterations.*

Proof. Let σ be a non-optimal strategy and σ^* an optimal MAX strategy. Since σ^* is not an anti-switch of σ, there is a vertex x such that $\sigma^*(x) \in IS_\sigma(x)$. We recall that $IS_\sigma(x)$ is the improvement set of x under σ defined Definition 3.15. We consider the policy iteration algorithm that finds such an x at each iteration and compute the σ-switch where x has been switched to $\sigma^*(x)$. Once a vertex x has been switched, for all strategy σ' computed after that, we have $\sigma^*(x) \notin IS_{\sigma'}(x)$, since $\sigma'(x) = \sigma^*(x)$. Thus, when a vertex is switched it will not be switched anymore in this algorithm. Hence, there are at most $|V_{\text{MAX}}| = n$ iterations. □

Notice that this does not require the SSG to be binary. Moreover, this policy is the fastest single switch policy since every mispositioned vertex needs to be switched, and we only switch those vertices.

Definition 4.3 (Hoffman-Karp algorithm)**.** *The Hoffman-Karp algorithm, also called total switch algorithm, is the policy iteration algorithm where every vertex of S_σ is switched towards the best option.*

In the rest of this section, we only consider SSG of degree 2. In their paper [TVK11], Tripathi, Valkanova and Kumar studied the total switch algorithm. They proved that this algorithm needs at most $O(2^n/n)$ iterations. This is based on two main results. The first one, stated in Proposition 4.5, gives a lower bound on the number of skipped strategies as a function of the size of the switch set at each step. The second one, that we present in Chapter 5, ensures that the algorithm only goes through a small number of strategies with a small switch set. First we introduce the concept of worst strategy.

Lemma 4.4. *For all game G, there exists a worst* MAX *strategy σ_0 such that for any* MAX *strategy σ, $\sigma \geq \sigma_0$.*

Proof. We consider the game G' which is a copy of G where every MAX vertex has been replaced by a MIN vertex. Then we consider the optimal strategy τ^* of G'. Let σ be the MAX strategy of G that play as in τ^* on V_{MAX}. Then for all strategy σ' and all vertex x $v_\sigma(x) \leq v_{\sigma'}(x)$. Otherwise, τ^* would not be optimal in G'. □

Proposition 4.5. *Let σ be a non-optimal* MAX *strategy, and let $\bar{\sigma}$ be the σ-total switch. There are at least $|S_\sigma| - 1$ strategies σ' different from $\bar{\sigma}$ such that $\sigma < \sigma' \leq \bar{\sigma}$.*

We offer a different proof than the one presented in [TVK11]. Note that the skipped strategies that we consider may not be the same as the strategies considered in the original proof. It is important to remember that we only consider SSG of degree 2.

Proof. Let σ be a strategy with switch set $S_\sigma = \{x_1, \ldots, x_k\}$. For $1 \leq i \leq k$, we define the strategy σ_i as the worst strategy, such that for $x \in V_{\text{MAX}} \setminus S_\sigma$, $\sigma_i(x) = \sigma(x)$ and $\sigma_i(x_i) \neq \sigma(x_i)$. Such a strategy exists by Lemma 4.4. This is not enough to guarantee that all those strategies are different, thus we add the following condition. If $S_\sigma \setminus S_{\sigma_i} \neq \{x_i\}$, then we can assume that for all j such that $x_j \in S_\sigma \setminus S_{\sigma_i}$, $\sigma_i(x_j) = \sigma(x_j)$. Indeed, since x_j is not in S_{σ_i}, changing the strategy in x_j results in an anti-switch of σ_i. Hence, such a strategy has value lesser or equal than σ_i and since σ_i is a worst strategy it has similar value and is also a worst strategy. Hence, if $\sigma_i = \sigma_j$, this would mean that x_j is in $S_\sigma \setminus S_{\sigma_i}$, but $\sigma_i(x_j) = \sigma(x_j) \neq \sigma_j(x_j)$. This ensures that each σ_i is distinct. By definition of the worst strategy, we have $\sigma_i \leq \bar{\sigma}$. Moreover, for all i, σ_i is a σ-switch and $\sigma_i > \sigma$. Since, $\bar{\sigma}$ can be in $\{\sigma_i \mid 1 \leq i \leq k\}$, there are $k - 1$ strategies $\sigma' \neq \bar{\sigma}$ such that $\sigma < \sigma' \leq \bar{\sigma}$ □

Those strategies will never be seen in the algorithm. This is a key argument to prove that there will be at most $O(2^n/n)$ iterations.

4.1.2 . Randomised Policy Iteration Algorithms

In [Con90], Condon presents a non-deterministic version of the Hoffman-Karp Algorithm for SSG of degree 2, where $2n$ σ-switches are chosen uniformly before selecting the best switch between them. They proved that the expected number of iterations of this algorithm is at most $2^{n-f(n)} + 2^{o(n)}$ for any function $f(n) = o(n)$. In [TVK11], Tripathi, Valkanova and Kumar also studied a non-deterministic policy iteration algorithm where at each step a σ-switch is uniformly chosen. They proved that with probability at least $1 - 2^{-2^{\Omega(n)}}$ their algorithm requires at most $O(2^{0.78n})$ iterations in the worst case.

Ludwig presented in [Lud95] a non-deterministic sub-exponential algorithm to solve SSG of degree 2 that we present in Algorithm 4. Notice that the strategy σ in input can be defined on a larger set of vertices.

Ludwig showed that this algorithm runs in an expected time of $O(2^{O(\sqrt{n})})$. Auger, Coucheney and Strozecki proved in [ACS19] that this algorithm can be reformulated as choosing uniformly an order on the MAX vertices and then choosing the switch vertex according to Bland's rule.

Algorithm 4: LudwigAlgorithm

Data: G a SSG of degree 2 and σ a MAX strategy
Result: A pair of optimal strategies (σ^*, τ^*)

begin
  if $V_{\text{MAX}} = \emptyset$ **then**
    Compute τ a best response to σ
    return (σ, τ)
  Choose uniformly at random a vertex x of V_{MAX}
  Construct G' the game where x is deleted and every edge (y, x) is replaced by an edge $(y, \sigma(x))$
  $(\sigma', \tau') \longleftarrow$ LudwigAlgorithm(G', σ)
  if (σ', τ') *is optimal* **then**
    return σ', τ'
  $\sigma'' \longleftarrow$ switch of σ' on x.
  Construct G'' the game where x is deleted and every edge (y, x) is replaced by an edge $(y, \sigma''(x))$
  return *LudwigAlgorithm*(G'', σ'')

4.2 . Lower Bound on Some Policy Iteration Algorithms

4.2.1 . Condon's Lower Bounds on Single Switch Policies

In [MC90], Melekopoglou and Condon give an exponential lower bound for several policy iteration algorithms with single vertex switch policy. We recall that we only focus on SSG of degree 2. We present in detail the lower bound of the ordered policy iteration given in Algorithm 5. We let $V_{\text{MAX}} = \{x_1, \ldots, x_n\}$.

Algorithm 5: OrderedPolicyIteration

Data: G a SSG of degree 2
Result: An optimal MAX strategy σ^*

begin
  $\sigma \longleftarrow$ a MAX strategy.
  while σ *is not optimal* **do**
    $i \longleftarrow$ largest index such that $x_i \in V_{\text{MAX}} \cap S_\sigma$.
    $\sigma \longleftarrow$ switch of σ on x
  return σ

The number of iterations of the ordered policy iteration algorithm is at most $2^n - 1$ since it is bounded by the number of MAX strategies and the optimal strategy loop is not visited. In fact, this upper bound can be reached.

Proposition 4.6 ([MC90])**.** *For every $n \geq 2$, there is an SSG such that the ordered policy iteration*

algorithm needs $2^n - 1$ *iterations to terminate.*

If the game used to prove the lower bound is the same as the one provided in [MC90], we give a different proof of the fact that $2^n - 1$ iterations are needed using Gray code. We recall that the Gray code is a representation in binary of integers, that use as many bits as their binary representation and that satisfy the fact that two successive values differ by only 1 bit. For more information on Gray code, we refer to [PTVF07, Knu11]. Some properties on Gray code require the XOR operator. We recall its definition.

Definition 4.7 (XOR)**.** *The bitwise XOR of two binary words* u *and* v *of same length* k *is the binary word of length* k w *where* $w_i = 0$ *if* $u_i = v_i$ *and* 1 *otherwise. We represent the bitwise XOR operator by* $\oplus$*.*

Thus, the bitwise XOR of two natural numbers x and y is the number whose binary representations is the bitwise XOR of the binary representation of x and y. For instance, the XOR of 22 and 14 is 24 since the binary representation of 22, 14 and 24 are respectively 10110_2, 01110_2 and 11000_2.

For any integer $k \leq 2^n - 1$, we call $\gamma^n(k)$ the Gray code of k on n bits and $b^n(k)$ the binary representation of k on n bits. The i^{th} components of $\gamma^n(k)$ is written $\gamma^n(k)_i$.

We recall that we have the following equalities.

$$\begin{aligned}
b^n(k)_0 &= \gamma^n(k)_0 \\
b^n(k)_1 &= \gamma^n(k)_0 \oplus \gamma^n(k)_1 \\
b^n(k)_2 &= \gamma^n(k)_0 \oplus \gamma^n(k)_1 \oplus \gamma^n(k)_2 \\
&\dots \\
b^n(k)_n &= \gamma^n(k)_0 \oplus \gamma^n(k)_1 \oplus \gamma^n(k)_2 \oplus \dots \oplus \gamma^n(k)_n
\end{aligned}$$

Here $\oplus$ is the bitwise operator XOR that we presented in Definition 4.7. We also have the equality $\gamma^n(k) = b^n(k) \oplus b^n(\lfloor k/2 \rfloor)$. Those results are presented for instance in [PTVF07, Knu11].

Proof. We consider the SSG G_n with $V_{\text{MAX}} = \{x_1, \dots, x_n\}$ and $V_R = \{a_1, \dots, a_n\}$ of same cardinality as V_{MAX}. This is a MDP. There are two sinks 0 and 1 that we sometimes denote by x_0 and a_0 for simplicity. Let $E = E_{\text{MAX}} \cup E_R$ with:

$$E_{\text{MAX}} = \{(x_i, x_{i-1}) \mid 1 \leq i \leq n\} \cup \{(x_i, a_i) \mid 1 \leq i \leq n\}$$

$$E_R = \{(a_i, a_{i-1}) \mid 1 \leq i \leq n\} \cup \{(a_i, x_{i-2}) \mid 2 \leq i \leq n\} \cup \{(a_1, 1)\}$$

with uniform probability distribution. We provide a figure of G_3 in Figure 4.1.

We consider G_n and the MAX strategy σ^b associated to a binary word b of n bits, such that $b^i = 1$ if and only if $\sigma^b(x_i) = a_i$. We can show that for any integer $k < n$, the value v of $\sigma^{\gamma(k)}$ satisfies

$$v(x_n) = \frac{\lceil \frac{k}{2} \rceil}{2^{n-1}}.$$

This is proven by induction on n. For $n = 1$ and $n = 2$, we can compute the two and four possible strategies and see that our property is true. We suppose that this is true for $n \geq 2$, and we prove that it is still true for $n + 1$. Recall that the binary representation of $\lfloor k/2 \rfloor$ correspond to removing the

last bit of the representation of k. This implies that if $\gamma^i(k)$ on i bits and $\gamma^{i+1}(k')$ on $i+1$ bits have same i first bits, then $b^i(k)$ and $b^{i+1}(k')$ also have same i first bits. Thus, the value of the vertices $x_1, \ldots, x_n$ can be computed using induction hypothesis. In the whole proof, we write $v_{k,n}$ the value vector of $v_{\sigma^{\gamma^n(k)}}$.

First, let us consider the case where $\gamma^{n+1}(k)_{n+1} = 0$. Then, on strategy $\sigma^{\gamma^{n+1}(k)}$ vertex x_{n+1} goes to x_n and thus $v_{k,n+1}(x_{n+1}) = v_{k,n+1}(x_n)$. We know that

$$\begin{aligned} v_{k,n+1}(x_n) &= v_{\lfloor k/2 \rfloor, n}(x_n) && \text{by only considering the first } n \text{ bits} \\ &= \frac{\left\lceil \frac{\lfloor k/2 \rfloor}{2} \right\rceil}{2^{n-1}} && \text{by induction hypothesis} \end{aligned}$$

Moreover, $\gamma^{n+1}(k) = b^{n+1}(k) \oplus b^{n+1}(\lfloor k/2 \rfloor)$. Hence, if $\gamma^{n+1}(k)_{n+1} = 0$ then either $b^{n+1}(k)_{n+1} = 0$ and $b^{n+1}(\lfloor k/2 \rfloor)_{n+1} = 0$, or $b^{n+1}(k)_{n+1} = 1$ and $b^{n+1}(\lfloor k/2 \rfloor)_{n+1} = 1$. This implies that either k can be divided by 4 or $k = 4k' + 3$. In the first case, we obtain

$$v_{k,n+1}(x_n) = \frac{k}{2^{n+1}} = \frac{\left\lceil \frac{k}{2} \right\rceil}{2^n}$$

In the second case we have:

$$v_{k,n+1}(x_n) = \frac{\left\lceil \frac{2k'+1}{2} \right\rceil}{2^{n-1}} = \frac{k'+2}{2^{n-1}} = \frac{\left\lceil \frac{4k'+3}{2} \right\rceil}{2^n}$$

Now, we consider the case where $\gamma^{n+1}(k)_{n+1} = 1$. In those strategies, vertex x_{n+1} goes to a_{n+1} and thus has value depending on $x_1, \ldots, x_{n-1}$. More precisely:

$$\begin{aligned} v_{k,n+1}(x_{n+1}) &= \frac{1}{2^n} + \sum_{i=1}^{n-1} \frac{v_{\lfloor k/2^{n+1-i} \rfloor, i}(x_i)}{2^{n-i}} \\ &= \frac{1}{2^n} + \sum_{i=1}^{n-1} \frac{\left\lceil \frac{\lfloor k/2^{n+1-i} \rfloor}{2} \right\rceil}{2^{n-i} \cdot 2^{i-1}} \end{aligned}$$

We show by induction that $\sum_{i=1}^{j} \left\lceil \frac{\lfloor k/2^{n+1-i} \rfloor}{2} \right\rceil = \lfloor k/2^{n+1-j} \rfloor$. We suppose that is true for j, then adding the term $\left\lceil \frac{\lfloor k/2^{n-j} \rfloor}{2} \right\rceil$ double the already obtained value $\lfloor k/2^{n+1-j} \rfloor$ and adds 1 if and only if $\lfloor k/2^{n-j} \rfloor$ is odd. Hence, it is $\lfloor k/2^{n+1-j} \rfloor$. This implies that:

$$v_{k,n+1}(x_{n+1}) = \frac{1}{2^n} + \frac{\lfloor k/2^2 \rfloor}{2^{n-1}}$$

Since $\gamma^{n+1}(k)n + 1 = 1$ it implies that $k = 4k' + \delta$ with $\delta \in \{1, 2\}$. Thus, we have both equalities:

$$\left\lceil \frac{k}{2} \right\rceil = 2k' + 1$$

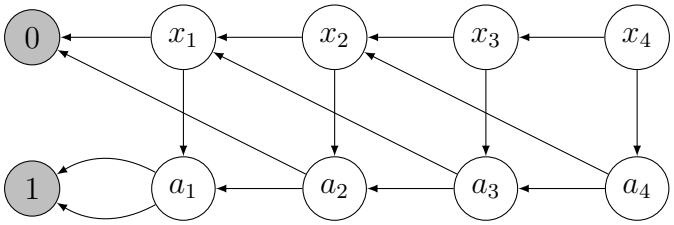


Figure 4.1: Representation of G_4

strategy	x_1	x_2	x_3	x_4
0000	0	0	0	0
0001	0	0	0	1/8
0011	0	0	1/4	1/8
0010	0	0	1/4	2/8
0110	0	1/2	1/4	2/8
0111	0	1/2	1/4	3/8
0101	0	1/2	2/4	3/8
0100	0	1/2	2/4	4/8
1100	1	1/2	2/4	4/8
1101	1	1/2	2/4	5/8
1111	1	1/2	3/4	5/8
1110	1	1/2	3/4	6/8
1010	1	1	3/4	6/8
1011	1	1	3/4	7/8
1001	1	1	1	7/8
1000	1	1	1	1

Table 4.1: The value of the MAX vertices of G_4

$$1 + 2\left\lfloor k'/4 \right\rfloor = 2k' + 1$$

Hence, we have proven that $v_{k,n+1}(x_{n+1}) = \dfrac{\left\lceil \frac{k}{2} \right\rceil}{2^n}$ which concludes our induction.

We provide a table of value of the strategy of G_4 in Table 4.1.

Thus, from the value we notice that for any $k < 2^n - 1$, $\sigma^{\gamma(k)} < \sigma^{\gamma(k+1)}$ and that $\sigma^{\gamma(k+1)}$ is a $\sigma^{\gamma(k)}$-switch where the switch vertex is the switchable vertex with the highest degree. Indeed, one can recall that incrementing a Gray code can be done in the following way. If the number of 1 is even, switch the last bit. Otherwise, switch the bit to the left to the rightmost 1. Moreover, if one switch the rightmost 1 or a bit to its right, one obtain the Gray code of a smaller integer. Hence, the ordered policy iteration visits every strategy $\sigma^{\gamma(k)}$ and thus terminates in exactly $2^n - 1$ iterations. □

Notice that Ludwig's randomised algorithm [Lud95] is simply the ordered policy iteration with an order randomly chosen. We notice that a_1 can be replaced by the sink 1, and thus the game has $n - 1$ random vertices. This gives a lower bound on GSIA for 2-SSG. If the strategy improvement method is not specified,

GSIA needs at least $2 \cdot 2^r - 1$ iterations on 2-SSG. This is not tight since our upper bound is of $n2^r$ iterations. This raises a number of open questions: can we obtain a lower bound of $n2^r$? Can the upper bound of GSIA be improved to $f(n)2^r$ with $f(n) = o(n)$? What is the lower bound for q-SSG? Note that for $r = 0$, it is possible to construct a game that needs $n + 1$ iterations. This is done by considering n MAX vertices x_i, edges $(x_1, 1)$, $(x_1, 0)$ and for $i \geq 2$, edges (x_i, x_{i-1}) and $(x_i, 0)$ and starting from strategy σ_0, defined for $1 \leq i \leq n$ as $\sigma_0(x_i) = 0$.

Melekopoglou and Condon also prove in [MC90] exponential lower bounds similarly for the following single vertex policy iteration methods.

1. Topological Policy Iteration: the vertex with more MAX vertices with a path to them is switched first. Then an arbitrary order is used to break ties.

2. Difference Policy Iteration: the vertex with the largest difference in value of its two out-neighbours is switched first.

3. Improvement Policy Iteration: the vertex with the largest difference between its value and its value when switched is switched first.

4.2.2 . Friedmann's Lower Bound for the Hoffman-Karp Algorithm

In [Fri09] and [Fri11] Friedmann proved an exponential lower bound on the Hoffmann-Karp algorithm. In order to do so, they proved the lower bounds on parity games. Then, they used the fact that parity games can be reduced to mean payoff game and discounted payoff game, which can be reduced to simple stochastic games [ZP96, HMZ13] as we have seen in Section 1.5.

The idea of the proof is to create a graph on which the total switch method will create a counter using cycles that are said to be either closed or opened. The technicalities are too heavy to be quickly explained here. Using $10k + 4$ vertices, they provide a graph that needs at least 2^k iterations before being solved by the Hoffmann-Karp algorithm.

This proof has been modified by Fearnley [Fea10] to create an exponential lower bound for the total switch policy for Markov Decision Processes.

4.3 . f-Strategies Algorithms

4.3.1 . GSIA and Gimbert and Horn Algorithm

The strategy improvement algorithm proposed by Gimbert and Horn in [GH08] (called GHA) can be viewed as an instance of GSIA where the set A of fixed arcs is the set R of all arcs going out of random vertices, and the improvement step in the subgame $G[R, \sigma]$ consists in taking an optimal strategy. In this case, the subgame $G[R, \sigma]$ is deterministic (random vertices are connected to sinks only and can be replaced by sinks), hence optimal values in $G[R, \sigma]$ depend only on the relative ordering of the values $v_\sigma(x)$ for sink and random vertices x of G. These values can be computed in $O(r \log(r) + n)$ time [AHMS08]. In the original paper [GH08], the algorithm is proposed in a context where the number of sinks is two, but we generalise their definitions to our context.

Definition 4.8 (f-strategy). *Consider a total ordering f on $V_R \cup V_S$, $f : x_1 < x_2 < \cdots < x_{r+s}$, where s is the number of sinks. An f-strategy corresponding to this ordering is an optimal* MAX

strategy in the game where the $s + r$ vertices above are replaced by sinks with new values satisfying $Val(x_1) < Val(x_2) < \cdots < Val(x_{r+s})$.

Clearly, this strategy does not depend on the actual values given but only on f. Note that if several f-strategies exist for a given f, they share the same values on all vertices.

The transformed game can be solved in quasi linear time $O(r \log(r) + n)$, as shown in [AHMS08]. The Algorithm GHA produces an improving sequence of f-strategies, and the two sinks of value zero and one are always first and last in the order, hence its number of iterations is bounded by $r!$, the total number of possible orderings of the random vertices. We extend this result to a large class of instances of GSIA: let us call Optimal-GSIA (Opt-GSIA), the meta algorithm obtained from Algorithm 3 with two additional constraints:

- the set A of fixed arcs is the same at each step of Algorithm 3;
- at line 5, the improving strategy σ' is the optimal strategy in $G[A, \sigma]$.

Theorem 4.9. *Consider an SSG G and a set of arcs A containing k arcs out of* MAX *or* MIN *vertices. Then, Algorithm Opt-GSIA runs in at most* $\min((r + k)q^r, (r + k)!)$ *iterations.*

Proof. Let σ be one of the iterated MAX strategies obtained by an instance of Opt-GSIA, and σ' be an optimal strategy in $G[A, \sigma]$. Then σ' is consequently an f-strategy in $G[A, \sigma]$, where f is the ordering on $V_R \cup V_A$ (where V_A is the set of A-sinks) which is induced by the value vector $v_{\sigma'}^{G[A,\sigma]}$ (if vertices have the same value, just arbitrarily decide their relative ordering in f).

Since strategies produced by the algorithm strictly increase in values by Proposition 3.9, they must be all distinct. Hence, the order f must be distinct at each step of the algorithm, which proves that Opt-GSIA does at most $(r + k)!$ iterations.

Moreover, at every step the value in G of at least one vertex in $V_R \cup V_A$ must improve by at least q^{-r} because of Theorem 2.1. Since the value of these vertices is bounded by 1, the number of iterations of Opt-GSIA is bounded by $(r + k)q^r$. □

4.3.2 . Generalised Gimbert and Horn Algorithm

Theorem 4.9 gives a competitive bound on the number of iterations for a strategy improvement algorithm, but the algorithmic complexity of an instance of Opt-GSIA also depends on how we find an optimal solution in $G[A, \sigma]$. We now present a class of instances of Opt-GSIA generalising GHA, with two interesting properties: there is a simple condition on A, which guarantees that $G[A, \sigma]$ is solvable in polynomial time, and they can be precisely compared to Ibsen-Jensen and Miltersen's algorithm (denoted by IJMA) [IJM12], which is the current best deterministic algorithm parametrized by r for 2-SSGs with a complexity of $O(r2^r(r \log(r) + |V|))$. We presented this algorithm in Section 2.3.2.

Let us describe IJMA, restated in our framework, and generalised to q-SSGs. IJMA is not a strategy improvement algorithm but a *value iteration algorithm*, which keeps a vector of values for random vertices, here denoted by v_i^{IJMA} at step i. This vector is updated in the following way:

- first, an optimal f-strategy is computed in the deterministic game $G[R, v_i^{\mathsf{IJMA}}]$, and we denote the values of this game by $v_i'^{\mathsf{IJMA}}$ (remember that R is the set of arcs going out of random vertices in G);

- second, the vector v_i^{IJMA} is updated on every random vertex by

$$v_{i+1}^{\mathsf{IJMA}}(x) = \sum_{y \in N^+(x)} p_x(y) \cdot v_i'^{\mathsf{IJMA}}(y).$$

As explained in Chapter 2, IJMA has an almost linear update complexity since $G[R, \sigma]$ is a deterministic game and can be solved in $O(r \log(r) + n)$ time.

Recall that Optimal-GSIA with R as a fixed set of arcs is equivalent to Gimbert and Horn's algorithm (GHA). When $A \subseteq R$ in Opt-GSIA, we obtain a generalisation of GHA, which can be precisely compared to IJMA as shown in the next theorem.

Theorem 4.10. *Opt-GSIA with $A \subseteq R$ needs fewer iterations than IJMA to find the optimal values on any input.*

Proof. We denote by σ_i the strategy obtained after i steps of an instance of Opt-GSIA, where $A \subseteq R$. We prove by induction on i that $v_{\sigma_i} \geq v_i^{\mathrm{IJMA}}$ (on random vertices). In IJMA, the value vector is initialised to 0 at the first step, hence any choice of initial strategy for Opt-GSIA guarantees a larger value on random vertices and satisfies the induction hypothesis.

Now assume that $v_{\sigma_i} \geq v_i^{\mathrm{IJMA}}$ for some i. First, we have

$$v_{\sigma_{i+1}} = v_{\sigma_{i+1}}^{G[A,\sigma_{i+1}]}$$

by Lemma 3.4 and

$$v_{\sigma_{i+1}}^{G[A,\sigma_{i+1}]} \geq v_{\sigma_{i+1}}^{G[A,\sigma_i]}$$

since $\sigma_{i+1} < \sigma_i$ by Proposition 3.9. Now $v_{\sigma_{i+1}}^{G[A,\sigma_i]}$ is, by definition of Opt-GSIA, an optimal value vector of $G[A, \sigma_i]$, but optimal values are larger in $G[A, \sigma_i]$ than in $G[R, \sigma_i]$ since $A \subseteq R$, and on the other hand, optimal values of $G[R, \sigma_i]$ are larger than those of $G[R, v_i^{\mathrm{IJMA}}]$, using Lemma 3.8 and the induction hypothesis, the latter being $v_i'^{\mathrm{IJMA}}$ by definition of IJMA. Putting these together, we have proven that

$$v_{\sigma_{i+1}} \geq v_i'^{\mathrm{IJMA}}.$$

Consider now a random vertex x. Considering the optimality conditions of Lemma 1.22 for $v_{\sigma_{i+1}}$ and the definition of $v_{i+1}^{\mathrm{IJMA}}(x)$, we see that

$$v_{\sigma_{i+1}}(x) = \sum_{y \in N^+(x)} p_x(y) v_{\sigma_{i+1}}(y) \qquad \geq \sum_{y \in N^+(x)} p_x(y) v_i'^{\mathrm{IJMA}}(y) = v_{i+1}^{\mathrm{IJMA}}$$

This concludes the induction and the result follows. □

We have proved that on every game, instances of Opt-GSIA with $A \subseteq R$ make less iteration than IJMA; and it can need dramatically less. Indeed, the analysis of IJMA [IJM12] relies on finding an extremal input for the algorithm, which happens to have no MAX nor MIN vertices. This extremal input is solved in one iteration of Opt-GSIA with the help of the "best response" step, and Opt-GSIA is then exponentially faster. We have yet no result to quantify how faster Opt-GSIA is in the general case, but we suspect the number of iterations of Opt-GSIA to be much smaller in many cases.

Corollary 4.11. *Opt-GSIA, with $A \subseteq R$ needs $O(2^r)$ iterations.*

Proof. Direct from the complexity of IJMA presented in Section 2.3.2. □

While the number of iterations of Opt-GSIA is better than the number of iterations of IJMA, one should take into account the complexity of a single iteration. In general, there is no algorithm for solving an iteration of Opt-GSIA in polynomial time, since when $A = \emptyset$ it is equivalent to solving any SSG; but let us consider a mild condition ensuring that it will be the case. Suppose that A contains at least one arc with transition probability at least $1/n$ out of every random vertex of G. This is not a very restrictive property, since there is at least one such arc for each random vertex, and thus at least 2^r sets A have this property. For q-SSGs, with q fixed, the condition can be simplified into saying that A contains at least an arc out of each random vertex. In this case, in the subgame $G[A, \sigma]$, there is a probability of stopping on a sink of at least $1/n$ when going through a random vertex. Hence, for this game, the value iteration algorithm (as presented in [IJM12]), converges in polynomial time in n to a value vector which is close enough to the optimal value vector so that we can recover it. Then, at the end of an iteration of Opt-GSIA, a best response must be computed, in time $O(rn^\omega)$. This should be compared to IJMA, whose iterations only requires an almost linear time. To remedy this, we propose a hybrid version between Opt-GSIA and IJMA that combines the good properties of both algorithms: do the same value iteration algorithm as IJMA, but once every $rn^{\omega-1}$ iterations, compute a best response, in time $O\left(rn^{\omega-1}\right)$, to update the values as in Opt-GSIA rather than doing a value propagation. This hybrid version enjoys the same complexity as IJMA since the overhead from the best response computation is in constant time per iteration. Moreover, the proof of Theorem 4.10 shows that it needs fewer iterations than IJMA. Moreover, in the extremal game of IJMA, that is the game on which IJMA is the slowest, Opt-GSIA needs exponentially fewer iterations.

4.4 . Condon's Converge from Below Algorithm

In [Con93], Condon first presents a faulty algorithm (the Naive Converge From Below Algorithm) and then a correct modified version, the Converge From Below (CFB) Algorithm. This algorithm proceeds by improving a value vector iteratively, but we show here that it is in fact a disguised strategy improvement algorithm, that can be seen as an instance of Opt-GSIA. This gives us a proof of convergence of the CFB algorithm in the general, non-stopping case (whereas Condon has the assumption that the game is stopping in her proof), and also bounds on the number of iterations (none are given in the original paper) by Theorem 3.20 and Theorem 4.9.

The CFB algorithm is restated with some clarifications on Algorithm 6 (we omit the details of the linear program, see [Con93]). The algorithm uses two properties of a vector, that we now define. First, a vector v is *feasible* if

1. For $s \in V_S$, $v(s) = \mathsf{Val}(s)$
2. For $r \in V_R$, $v(r) = \sum_{x \in N_r} p_r(x) v(r)$
3. For $x \in V_{\text{MIN}}$, $v(x) \leq \min_{y \in N^+(x)} v(y)$
4. For $x \in V_{\text{MAX}}$, $v(x) \geq \max_{y \in N^+(x)} v(y)$.

A feasible vector is *stable* at x a MIN vertex (resp. MAX vertex) if it satisfies condition 3 (resp. condition 4) of feasibility for x with an equality.

We now show by induction that the CFB algorithm is equivalent to the instance of Opt-GSIA where all MIN vertices are fixed, i.e. A is the set of arcs entering MIN vertices. Let A_{MIN} denote this set.

To see this, suppose that at the beginning of line 5 of CFB, Vector v_r is the value vector of a MAX-strategy σ in G. Then:

- at Line 5, we "update v as the feasible vector where all MIN vertices x have value $v_r(x)$ and all MAX vertices are stable". This amounts to finding a MAX-strategy σ' which satisfies optimality conditions in $G[A_{\text{MIN}}, \sigma]$, i.e. an optimal strategy for MAX in this subgame. This is exactly the subgame improvement step of Opt-GSIA. At the end of this step, v is the optimal value vector in $G[A_{\text{MIN}}, \sigma]$;
- in the next loop, at Line 4 of CFB, we "compute the value vector v_r of an optimal response to the MAX strategy that plays greedily according to v", i.e. v_r is updated to the value vector $v^G_{\sigma'}$. This is precisely Line 6 of GSIA when we update values in the subgame.

Hence, we see that, except for the initialisation where v_r may not correspond to a MAX-strategy, it will be the case as soon as we reach Line 5 of the first loop, and from this point on CFB will correspond exactly to the instance of Opt-GSIA described above.

Algorithm 6: Converge From Below Algorithm

Data: G an SSG
Result: The optimal value vector v^* of G

```
1 begin
2     · let v be a feasible vector in which all MIN vertices have value 0 and all MAX
        vertices are stable
3     while v is not an optimal value vector do
4         · use linear programming to compute the value vector v_r of an optimal response
            to the MAX strategy that plays greedily according to v
5         · update v as the feasible vector where all MIN vertices x have value v_r(x) and
            all MAX vertices are stable
6     return v
```

4.5 . New Algorithms from GSIA

We can use GSIA to design many strategy improvement algorithms. We present three of them, all based on a choice of A which makes $G[A, \sigma]$ solvable in polynomial time. The initial strategy can be anything and σ' is always chosen to be the optimal strategy in $G[A, \sigma]$. Most of them can be seen as generalisations of known algorithms.

1. Let A be a feedback arc set of G: a set of edges such that if removed make the game acyclic, then $G[A, \sigma]$ is acyclic, and it can be solved in linear time. It seems intuitively appealing to think that

this algorithm will be faster if the feedback arc set is small, but we have no idea how to prove such a proposition.

2. A MAX acyclic SSG is an SSG such that every MAX vertex has at most one outgoing arc in a cycle. MAX acyclic SSG can be solved in polynomial time, see [ACS14]. If we let A be a set of arcs that contains all but one outgoing arcs of each MAX vertex, then $G[A, \sigma]$ is MAX acyclic and can be solved in polynomial time. Moreover, such a game can be solved by strategy improvement in at most n iterations. This can be seen as a generalisation of Hoffman-Karp algorithm, in which A contains *all* outgoing arcs of MAX vertices.

3. As an intermediate between acyclic games and MAX acyclic games, we may consider almost acyclic games, where all vertices have at most one outgoing arc in a cycle. Almost acyclic SSGs can be solved in linear time [ACS14].

4.6 . Choosing the Initial Strategy: Dai and Ge Algorithm

In [DG09], Dai and Ge give a randomised improvement of GHA simply by choosing a better initial strategy. To do so, they choose randomly $\sqrt{r!}\log(r!)$ f-strategies and choose the one whose sum of values hold the highest value. This can also be done using the max of two strategies that we introduced in Section 3.4. This ensures, with high probability, that at most $\sqrt{r!}$ iterations will be done in GHA. Thus, their algorithm runs in $O\left(\sqrt{r!}\right)$ iterations. This algorithm is also captured by GSIA by selecting the initial strategy in the same way. However, it seems hard to combine the gain made by the random selection of the strategy and the bound in $O(q^r)$ of GSIA, since even a strategy close to the optimal one may have values far from it. Remark that it is trivial to extend this method to any instance of Opt-GSIA to improve on the complexity of Theorem 4.9. With the same demonstration as in [DG09], by first selecting $\sqrt{(r+k)!}\log((r+k)!)$ f-strategies randomly and using the one with the highest sum as a starting strategy, with high probability Opt-GSIA will need at most $\sqrt{r!}$ iterations.

5 - Recursive Algorithms

Contents

In this chapter, we study recursive strategy iteration algorithms. The general idea is to fix the strategy on some vertices, then optimally solving the rest of the game. We have independently found the algorithms that we present in [BdM21] but a reviewer mentioned that these algorithms are similar to ones used for solving USO. We still present them since they are not well known to the SSG community, and we hope that they might be improved for the specific case of SSGs. Moreover, they are strategy improvement algorithms and do not require the game to be stopping. Even if their complexity is better than the complexity of the algorithm presented in Chapter 4, it is worse than the complexity of USO algorithms presented in Chapter 6.

5.1 . Properties on Switches

5.1.1 . Fixed Strategy Subgame

In order to further study switch strategies, we present the fixed strategy subgame that we introduced in [BdM21]. For T a set of MAX vertices, and σ a MAX strategy, we define the subgame $G_{|T[\sigma]}$ as the game G where the MAX vertices x of T have been replaced with random vertices that go to $\sigma(x)$ with probability one. In other words, $G_{|T[\sigma]}$ is the game G where MAX has to play as in σ in T.

Definition 5.1. *Let $G = (V, E)$ an SSG. For σ a strategy and T a subset of V_{MAX} the fixed strategy subgame that we write $G_{|T[\sigma]} = (V, E')$ is the SSG that is a copy of G where $E' = E \setminus \{(x, y) \mid x \in T,\ y \neq \sigma(x)\}$ and all vertices x of T are transformed in random vertices, with associated probability distribution $p_x(\sigma(x)) = 1$.*

We provide an example of this transformation in Figure 5.1 and in Figure 5.2.

There is a bijection between the strategies of $G_{|T[\sigma]}$ and the strategies of G that play as in σ in T. In the rest of the chapter, we identify those two sets of strategies. Moreover, for a strategy σ' that plays as in σ in T and any MIN strategy τ, there is equality between the value vectors $v_{\sigma',\tau}$ in G and $G_{|T[\sigma]}$.

This framework allows us to restate and to provide simple and elegant proofs to some known result on SSGs.

Lemma 5.2 ([TVK11])**.** *For any MAX strategy σ, σ is optimal in $G_{|S_\sigma[\sigma]}$.*

Proof. The switch set of σ in $G_{|T[\sigma]}$ is empty, hence, by Corollary 3.19, σ is optimal in $G_{|T[\sigma]}$. □

5.1.2 . Switch Set and Switch Vector

One of the first properties of the switch set is the bijection between value vector and switch set for simple stochastic game of degree 2. This result is used by Tripathi, Valkanova and Kumar in [TVK11] in

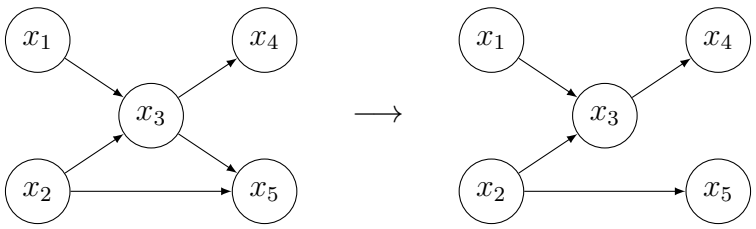


Figure 5.1: Transformation of the graph G in $G_{|\{x3\}[\sigma]}$ where $\sigma(x_3) = x_4$

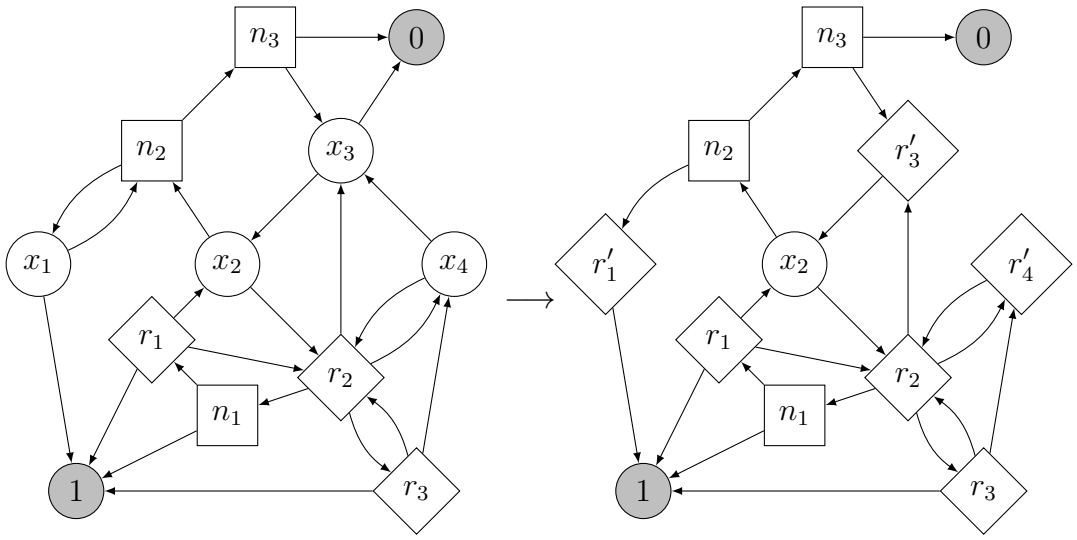


Figure 5.2: Transformation of the game G in the game $G_{|T[\sigma]}$ where $T = \{x_1; x_3, x_4\}$ and $\sigma(x_1) = 1$, $\sigma(x_3) = x_2$ and $\sigma(x_4) = r2$. The probability distribution on the random vertices is the uniform distribution.

order to obtain the bound of $O\left(2^n/n\right)$ iterations of their algorithm. In this section, we will also see that this result can be extended to SSG of larger degree.

First, we focus on SSG of degree 2. The following proposition also appears in [TVK11] and it allows us to study strategies by just looking at their switch set.

Proposition 5.3. *Let G be an SSG of degree 2. For σ and σ' two* MAX *strategies such that $S_\sigma \subsetneq S_{\sigma'}$, then $v_\sigma > v_{\sigma'}$.*

Proof. Let σ and σ' be two MAX strategies such that $S_\sigma \subsetneq S_{\sigma'}$. If σ' is a strategy of $G_{|S_\sigma[\sigma]}$, then σ' is not optimal in $G_{|S_\sigma[\sigma]}$ by Corollary 3.19, and by Lemma 5.2, $v_\sigma > v_{\sigma'}$. Otherwise, we consider σ'' that plays as in σ' in $V_{\text{MAX}} \setminus S_\sigma$ and as in σ in S_σ. Strategy σ'' is a σ'-switch, and a strategy of $G_{|S_\sigma[\sigma]}$, thus: $v_{\sigma'} < v_{\sigma''} \leq v_\sigma$. □

In the same way, we can also consider the case where both switch sets are equal.

Lemma 5.4. *Let G be an SSG of degree 2. For σ and σ' two* MAX *strategies such that $S_\sigma = S_{\sigma'}$, then $v_\sigma = v_{\sigma'}$ and for every x in S_σ, $\sigma(x) = \sigma'(x)$*

Proof. Let σ and σ' be two strategies that satisfy the hypothesis and for the sake of contradiction, let us assume that there is x in S_σ such that $\sigma(x) \neq \sigma'(x)$. Then, the strategy σ'' that plays as σ in S_σ and as σ' in the other vertices is a σ'-switch. Thus $v_{\sigma''} > v_{\sigma'}$. Since σ'' plays as σ on S_σ, by Lemma 5.2, $v_\sigma \geq v_{\sigma''}$. Thus $v_\sigma > v_{\sigma'}$. We similarly can prove that $v_{\sigma'} > v_\sigma$, which yields a contradiction. Hence, σ and σ' plays similarly on S_σ. The strategy σ and σ' are optimal strategy of $G_{|S_\sigma[\sigma]}$ and thus have the same value. □

The number of possible switch sets is at most 2^n, while the number of MAX strategies for SSG of degree d is d^n. Hence, looking at the switch set is not enough to characterise strategies for SSG of higher degree. However, it is possible to only look at the size of the Improvement Set as defined in Definition 3.15.

Definition 5.5 (Switch vector)**.** *The switch vector $\vec{S_\sigma}$ of a strategy σ is a vector of dimensions $|V_{\text{MAX}}|$ such that for $x \in V_{\text{MAX}}$, $\vec{S_\sigma}(x) = |IS_\sigma(x)|$.*

Informally, for a MAX strategy σ, the switch vector counts for each MAX vertex x the number of neighbours of x with a better value under σ. Like value vectors, we compare switch vectors using the pointwise order.

Theorem 5.6. *Let G be an SSG. For two strategies σ and σ' such that $\vec{S_\sigma} < \vec{S_{\sigma'}}$ we have $\sigma > \sigma'$.*

Proof. In order to prove this result, we construct a strategy σ'' that can be equal to σ such that $\sigma' < \sigma'' \leq \sigma$. We recall that for all x, $\sigma(x) \notin IS_\sigma(x)$. First, for all x such that $\vec{S_\sigma}(x) < \vec{S_{\sigma'}}(x)$ there exists y in $IS_{\sigma'}(x)$ that is not in $IS_\sigma(x)$. We define $\sigma''(x) = y$. If $\vec{S_\sigma}(x) = \vec{S_{\sigma'}}(x)$ then, either $\sigma'(x) \notin IS_\sigma(x)$ and we define $\sigma''(x) = \sigma'(x)$, or $\sigma'(x) \in IS_\sigma(x)$ and thus there is y in $IS_{\sigma'}(x)$ that is not in $IS_\sigma(x)$ and we define $\sigma''(x) = y$. Thus, σ'' is a σ'-switch. Moreover, either σ'' is equal to σ or σ'' is as σ-anti-switch and in both cases, $\sigma'' \leq \sigma$. Thus, by Lemma 3.18, $\sigma' < \sigma'' \leq \sigma$ and we have proven that $\sigma > \sigma'$. □

To the best of our knowledge this characterisation is not known, and we hope that it could lead to interesting algorithms in the case of SSG of degree d.

5.1.3 . Super-Switch

In this section, we extend the classical notion of switch by following it by the computation of an optimal strategy on some of its vertices.

Definition 5.7. *Let σ be a MAX strategy with a non-empty switch set S_σ and $T \subset V_{\text{MAX}}$ with $T \cap S_\sigma \neq \emptyset$. A (σ, T)-super switch is defined as an optimal strategy in $G_{|T[\sigma']}$, where σ' is a switch of σ that switch at least one vertex of T.*

A (σ, T)-super-switch is a strategy $\tilde{\sigma}$ obtained from an intermediate σ-switch σ' such that $\forall x \notin T$, $\sigma(x) = \sigma'(x)$, by computing an optimal strategy of $G_{|T[\sigma']}$. We give a representation of a super-switch in Figure 5.3

Lemma 5.8. *For σ a non-optimal MAX strategy, T a set of MAX vertices and σ' a (σ, T)-super-switch, $v_{\sigma'} > v_\sigma$.*

Proof. We consider the strategy σ'' that plays as σ' on T and as σ on $V_{\text{MAX}} \setminus T$. The strategy σ'' is a σ-switch and by Lemma 3.18 $v_{\sigma''} > v_\sigma$. Moreover, σ' is the optimal strategy of $G_{|T[\sigma'']}$ and σ'' is also a strategy of $G_{|T[\sigma'']}$. Hence, $v_{\sigma'} \geq v_{\sigma''}$ and $v_{\sigma'} > v_\sigma$. □

As we stated in Chapter 4, Tripathi, Valkanova and Kumar, show in [TVK11] that for a strategy σ, there is at least $|S| - 1$ σ-switch σ' different from $\bar{\sigma}$ such that $v_{\sigma'} \leq v_{\bar{\sigma}}$. In Theorem 5.9 we adapt their proof to show that this result can be extended to super-switch.

For σ a MAX strategy, T a set of MAX vertices and σ' a (σ, T)-super-switch, we denote by $D_{\sigma,\sigma'}$ the set $\{x \mid \sigma(x) \neq \sigma'(x), x \in T\}$ the set of vertices that has been switched in the intermediate σ-switch.

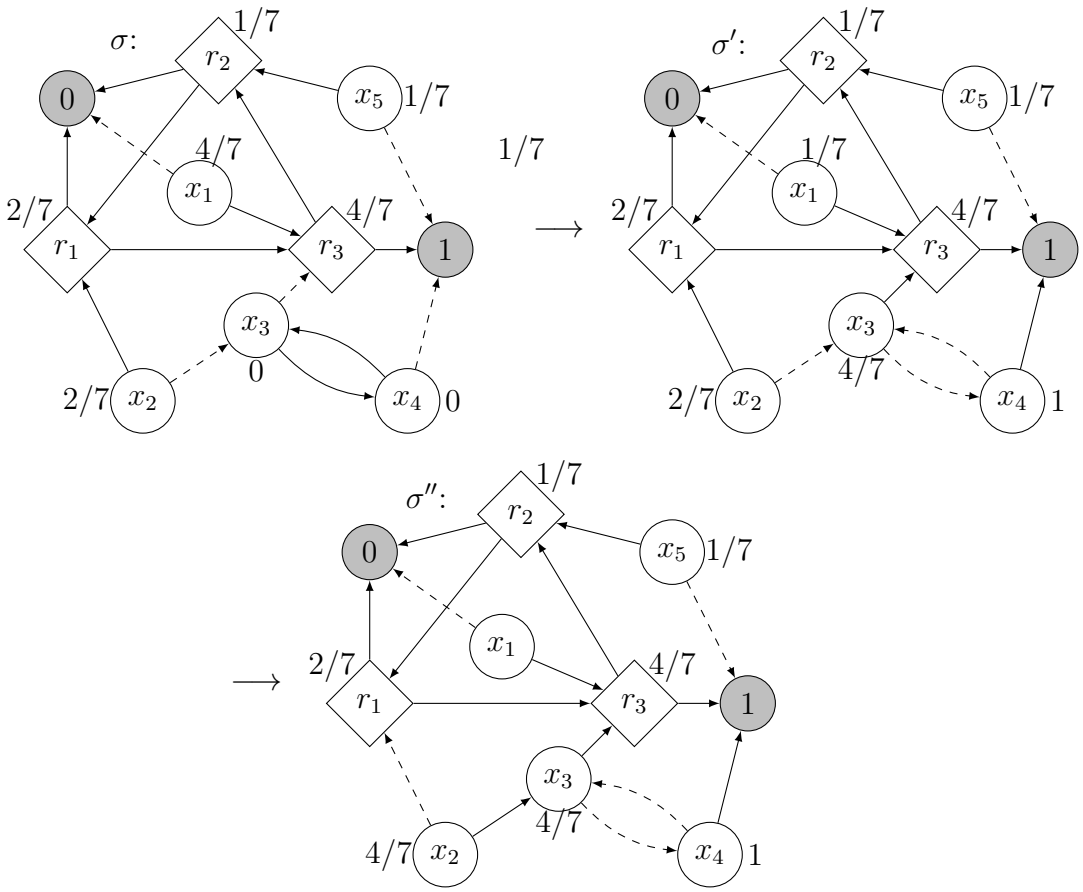


Figure 5.3: The strategy of MAX are represented by plain arcs and the probability distribution on the random vertices is the uniform distribution. The switch set of σ, S_σ is $\{x_3, x_4, x_5\}$. The strategy σ' is a σ-switch and σ'' is a (σ, S_σ)-super-switch.

Theorem 5.9. *For σ a non-optimal* MAX *strategy, T a set of* MAX *vertices intersecting S_σ and σ' a (σ, T)-super-switch, there is at least $|D_{\sigma,\sigma'}| - 1$ (σ, T)-super-switches $\sigma'' \neq \sigma'$ such that $v_\sigma < v_{\sigma''} \leq v_{\sigma'}$.*

Proof. First, we show that Theorem 5.9 is true in the case $D_{\sigma,\sigma'} = 2$. Let $D_{\sigma,\sigma'} = \{x, y\}$. We consider the game G' which is the game $G_{|T \setminus \{x,y\}[\sigma]}$ where all edges (x, x') and (y, y') with $x' \notin \{\sigma(x), \sigma'(x)\}$ and $y' \notin \{\sigma(y), \sigma'(y)\}$ have been removed. Both σ and σ' are strategies in the game G'. We denote by $S_{\tilde{\sigma}}$ the switch set of a strategy $\tilde{\sigma}$ in the game G and $S'_{\tilde{\sigma}}$ the switch set of a strategy $\tilde{\sigma}$ in the game G'. Hence, $S'_\sigma = \{x, y\}$. Let us call σ_x and σ_y the (σ, T)-super-switch where respectively only x and only y has been switched in T.

By Lemma 5.3, $S'_{\sigma'}$ is strictly included in $\{x, y\}$ and thus is at most a singleton. If $S'_{\sigma'}$ is empty, then, $v_\sigma < v_{\sigma_x}, v_{\sigma_y} \leq v_{\sigma'}$. We suppose that $S'_{\sigma'} = \{x\}$. Then, we notice that σ_x and σ' are both strategies of $G'_{|\{x\}[\sigma']}$ and σ' is optimal in $G'_{|\{x\}[\sigma']}$. Thus, $v_\sigma < v_{\sigma_x} \leq v_{\sigma'}$ and $v_{\sigma_y} > v_{\sigma'}$. This implies that if $v_{\sigma_x} \not> v_{\sigma_y}$, $v_{\sigma'} \geq v_{\sigma_x}$ and if we also have $v_{\sigma_x} \neq v_{\sigma_y}$ then $y \in S'_{\sigma_x}$

Let us consider the general case with $D_{\sigma,\sigma'} = \{x_1, \ldots, x_t\}$. For $E \subset \{1, \ldots, t\}$, we write v_E the value of the (σ, T)-super-switch $\sigma_{\{E\}}$ where only the vertices x_i for $i \in E$ has been switched. We assume that for all $i < j \leq t$, $v_{\{i\}} \not> v_{\{j\}}$. If for all $j > 1$, $v_{\{1\}} \neq v_{\{j\}}$ then $v_{\{1,j\}} \geq v_{\{1\}}$ and $j \in S_{\sigma_{\{1\}}}$.

Otherwise, we suppose that there is $k > 1$ such that for all $2 \leq j \leq k$, $v_{\{1\}} = v_{\{j\}}$ and for all $j > k$, $v_{\{1\}} \neq v_{\{j\}}$. We consider the game G' which is the game $G_{|T \setminus \{x_1,\ldots,x_k\}[\sigma]}$ and where for $i \leq k$, all edges from x_i not towards $\sigma(x_i)$ or $\sigma'(x_i)$ are removed. We denote by S' the switch sets in G' and $\sigma' = \sigma_{\{1,\ldots,k\}}$. By Lemma 5.3, $S'_{\sigma'}$ is strictly included in $\{1, \ldots, k\}$. We suppose that $S'_{\sigma'} = \{1, \ldots, k'\}$ for some $k' < k$. By induction hypothesis, we know that $v_{\{1\}} \leq v_{\{1,\ldots,k'\}}$ and σ' and $\sigma_{\{1,\ldots,k'\}}$ are both optimal strategies of $G'_{|\{1,\ldots,k'\}[\sigma']}$ and, thus, have the same value. Thus, $v_E \leq v_{\sigma'}$ for all $E \subseteq \{1, \ldots, k\}$ and for all $j > k$, $x_j \in S_{\sigma'}$. We conclude by induction on $\{x_j, \ldots\}$.

Thus, we have that $v_{\sigma'} = v_{\{1,2,\ldots,t\}} \geq v_{\{1,2,\ldots,t-1\}} \geq \ldots \geq v_{\{1\}}$ and we have proven that there is at least $|D_{\sigma,\sigma'}| - 1$ (σ, T)-super-switches $\sigma'' \neq \sigma'$ such that $v_\sigma < v_{\sigma''} \leq v_{\sigma'}$. □

We notice that a σ-switch is a (σ, V_{MAX})-super-switch. Hence, Theorem 5.9 also proves the main theorem of [TVK11].

5.2 . A Recursive Algorithm on Pairs of Vertices

In all this section, we consider SSGs of degree d.

Algorithm 7 works by fixing the strategy of the game on two vertices, then recursively solving the rest of the game. If this does not yield an optimal strategy, then it switches the strategy on the fixed vertices and iterates. The switch sets of the considered strategies after the recursive call (line 9) are included in $\{x, y\}$.

Let us first recall that computing v_σ can be done in polynomial time in $|G|$ by solving a linear programming problem.

We write N the number of iterations of the loop. Let σ_i be the value of σ at the start of the i^{th} iteration of the loop line 8 and σ_{N+1} the value of σ after the last iteration of the loop. Algorithm 7 makes $N + 1$ recursive calls to an instance with $n - 2$ MAX vertices. We notice that for all i, $S_{\sigma_i} \subseteq \{x, y\}$.

Lemma 5.10. *For all $i < N + 1$, $|S_{\sigma_i}| \neq 0$. Moreover, there are at most $2(d - 1)$ indices k such that $|S_{\sigma_k}| = 1$.*

Algorithm 7: RecursivePair

Data: G an SSG
Result: σ an optimal MAX strategy and v the optimal value vector.

```
begin
    if |V_MAX| ≤ 1 then
        Compute the optimal strategy σ by testing all possibilities
        return (σ, v_σ)
    σ ⟵ a MAX strategy
    x, y ⟵ two vertices of V_MAX
    (σ, v) ⟵ RecursivePair(G_{|{x,y}[σ]})
    while σ is not optimal do
        σ ⟵ σ̄
        (σ, v) ⟵ RecursivePair(G_{|{x,y}[σ]})
    return (σ, v)
```

Proof. If $S_{\sigma_i} = \emptyset$, then the algorithm stops and $i = N + 1$. Thus, for all $k < N + 1$, $S_{\sigma_k} \neq \emptyset$.

For all neighbours x' of x, there is at most one k such that $\sigma_k(x) = x'$ and $S_{\sigma_k} = \{x\}$ since such strategies are all optimal in $G_{|\{x\}[\sigma_k]}$ and thus have the same value. Moreover, if there is an optimal strategy σ^* such that $\sigma^*(x) = x'$, then all optimal strategies of $G_{|\{x\}[\sigma^*]}$ are optimal on G and there is no strategy σ such that $\sigma(x) = x'$ and $S_\sigma = \{x\}$. Hence, there is at most $2(d-1)$ visited strategies with switch set of size 1. □

We can now notice that when $|S_{\sigma_i}| = 2$, then a super-switch is skipped by Theorem 5.9, and thus this skips one call to an instance with $n - 2$ MAX vertices. This allows us to gives a bound on the complexity of Algorithm 7.3

Proposition 5.11. *Algorithm 7 runs in* $O\left(\left(\left\lfloor \frac{(d+1)^2}{2} \right\rfloor - 1\right)^{n/2} Poly(|G|)\right)$.

Proof. If we write, n_0, n_1 and n_2 the number of indices i such that σ_i has a switch set of respectively size 0, 1 and 2, then $N + 1 = n_0 + n_1 + n_2$. By Theorem 5.9, if $S_{\sigma_i} = \{x, y\}$, then there is a super-switch σ' such that $v_{\sigma_i} < v_{\sigma'} \leq v_{\sigma_{i+1}}$. Thus, $n_0 + n_1 + 2n_2 \leq d^2$. We also know, by Lemma 5.10, that $n_0 + n_1 \leq 2d - 1$. Then:

$$2(N + 1) = (n_0 + n_1) + (n_0 + n_1 + 2n_2) \leq d^2 + 2d - 1$$

Which gives

$$N + 1 \leq \frac{(d+1)^2}{2} - 1$$

Since $N + 1$ is an integer, we have $N + 1 \leq \left\lfloor \frac{(d+1)^2}{2} - 1 \right\rfloor$. Hence, Algorithm 7 makes at most $\left\lfloor \frac{(d+1)^2}{2} - 1 \right\rfloor$ recursive calls to an instance with $n - 2$ MAX vertices and Algorithm 7 runs in

$$O\left(\left(\left\lfloor\frac{(d+1)^2}{2}\right\rfloor-1\right)^{n/2}Poly(|G|)\right).$$ □

In the case of SSG of degree 2, Algorithm 7 is similar to Ludwig's Algorithm [Lud95] which fixes the strategy on the vertices one at a time. The choice of which vertex to fix is random and provides an algorithm that runs in expected time $2^{O(\sqrt{n})}Poly\left(|V|\right)$. However, despite the similarity of the two algorithms, we were yet not able to find a similar analysis as the one in [Lud95] to the stochastic version of Algorithm 7. Notice that those complexity bounds are worse than the ones for USO that we present in Chapter 6.

On binary SSG, Algorithm 7 gives a complexity bound in $O\left(\sqrt{3}^n Poly(|G|)\right)$, which is better than the one for binary SSG in [TVK11]. However, it is still possible to improve this complexity, as we show in the next section.

5.3 . A Recursive Algorithm for Binary SSG

In all this section, we will only consider binary SSG.

5.3.1 . The Algorithm

The idea of Algorithm 8 is to fix a subset of vertices and recursively solve the rest of the game. Then, we switch the current strategy and fix a smaller subset of vertices and reiterate. We show that we never make a call to an instance with $n-1$ MAX vertices. This is done by carefully selecting the set of fixed vertices. While being found independently, this algorithm is very close to the Fibonacci Seesaw Algorithm of Szabó and Welzl for USO [SW01] that we present in Chapter 6. It nonetheless has the advantage to be a strategy improvement algorithm and to work on non-stopping SSG.

Algorithm 8: DecreasingFixedSet

```
Data: G an SSG
Result: σ an optimal MAX strategy
1  begin
2      σ ⟵ a MAX strategy
3      S ⟵ S_σ
4      σ ⟵ σ̄
5      T ⟵ S ∪ S_σ
6      S ⟵ S_σ
7      while S ≠ ∅ do
8          σ ⟵ σ̄
9          σ ⟵ DecreasingFixedSet(G_{|T[σ]})
10         T ⟵ S ∪ S_σ
11         S ⟵ S_σ
12     return (σ, v)
```

As stated before, the goal of Algorithm 8 is to avoid the call to a game with $n-1$ MAX vertices. In order to achieve this, the set of vertices that is fixed in the recursive call is the union of the previous and

current switch set.

Lemma 5.12. *Algorithm 8 terminates and computes an optimal* MAX *strategy and its value vector.*

Proof. The value vectors of the visited strategies are increasing by Lemma 3.18 and there are a finite number of MAX strategies, hence Algorithm 8 terminates. The algorithm ends when the switch set of a MAX strategy is empty, hence when the algorithm terminates it computes an optimal MAX strategy. Alternatively, Algorithm 8 is a strategy improvement algorithm, thus by Theorem 3.13, it terminates and computes an optimal strategy. □

5.3.2 . Time Complexity

Let us call $T_0 = V_{\text{MAX}}$ and S_0 and σ_0 the value of the variables S and σ after line 3. In addition, we call σ_i, T_i and S_i the value of the variables σ, T and S at the beginning of the i^{th} iteration of the while loop. Let N be the number of iterations. We call T_{N+1}, S_{N+1} and σ_{N+1} the value of those variables at the end of the last iteration. By line 11 of Algorithm 8 for every i, $S_i \subset T_i$. We create a partition of T_i by considering S_i and $S_{i-1} \setminus S_i$.

If, for all $i \geq 1$, $S_{i-1} \setminus S_i$ is not empty, then $|T_i| > |S_i|$ and S_i not empty implies that $|T_i| \geq 2$. Then, all recursive calls to Algorithm 8 are made to a subgame with at most $n-2$ MAX vertices.

Proposition 5.13. *For every $1 \leq i \leq N$, $S_{i-1} \setminus S_i$ is not empty.*

Proof. The strategy σ_i is a (σ_{i-1}, T_{i-1})-super-switch, thus $v_{\sigma_i} > v_{\sigma_{i-1}}$. Hence, by contraposition of Proposition 5.3, S_{i-1} is not a subset or equal to S_i. □

Now, we need to prove that each iteration of the loop strictly decreases the size of T. In order to better visualise the proof of the following proposition, a representation of the successive switches is provided in Figure 5.4.

Proposition 5.14. *For $1 \leq i < N$, $T_{i+1} \subsetneq T_i$.*

Proof. Let $1 \leq i < N$. We notice that $S_i \subseteq T_{i-1}$, since σ_i is optimal in $G_{T_{i-1}[\sigma_i]}$, and $T_i = S_{i-1} \cup S_i$. Thus, we have $T_i \subseteq T_{i-1}$.

We recall that strategy σ_{i-1} is optimal in the game $G_{S_{i-1}[\sigma_{i-1}]}$. We notice that for every $x \in S_{i-1} \cap S_i$, $\sigma_{i+1}(x) = \sigma_{i-1}(x)$: the strategy of every vertex in $S_{i-1} \cap S_i$ has been changed twice, thus going back to its original value since we consider binary SSG. We recall that $S_{i-1} \setminus S_i$ is not empty by Proposition 5.13 and assume for the sake of contradiction that $S_{i-1} \setminus S_i \subset T_{i+1}$. Since $T_{i+1} = S_i \cup S_{i+1}$, this implies that $S_{i-1} \setminus S_i \subset S_{i+1}$. Then we define the strategy σ' as follows:

$$\forall x \in S_{i-1} \setminus S_i,\ \sigma'(x) \neq \sigma_{i+1}(x)$$

$$\forall x \notin S_{i-1} \setminus S_i,\ \sigma'(x) = \sigma_{i+1}(x)$$

Since $S_i \setminus S_{i-1}$ is a subset of S_{i+1}, σ' is a σ_{i+1}-switch and by Lemma 3.18 $\sigma' > \sigma_{i+1} > \sigma_i$. However, for all $x \in S_{i-1}$, $\sigma'(x) = \sigma_{i-1}(x)$ and σ' is a strategy of $G_{S_{i-1}[\sigma_{i-1}]}$ which contradicts the optimality of σ_{i-1} on this game. This shows that there exists x in $S_{i-1} \setminus S_i$ but not in S_{i+1}. In other words, **there is x in S_{i-1} and thus in T_i but not in $S_i \cup S_{i+1}$ and thus not in T_{i+1}**. Therefore, we have proven that $T_{i+1} \subsetneq T_i$. □

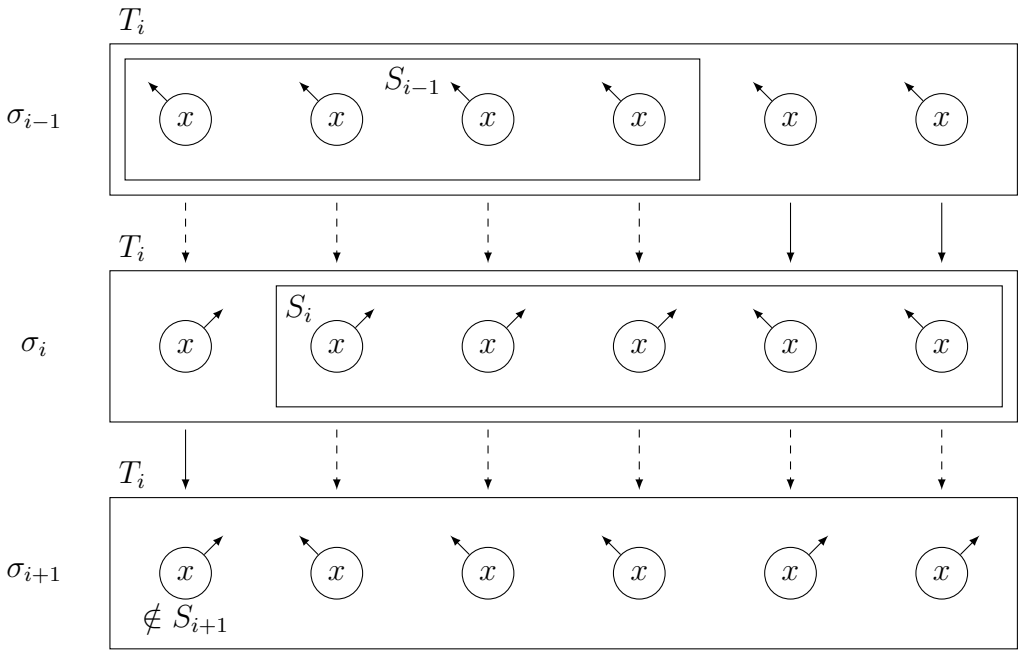


Figure 5.4: Strategy on the vertices of T_i under strategies σ_{i-1}, σ_i and σ_{i+1}

We can now give a bound on the complexity of Algorithm 8.

Theorem 5.15. *Algorithm 8 has time complexity* $O(\varphi^n Poly(|G|))$*, where* $\varphi = \dfrac{1+\sqrt{5}}{2}$ *is the golden ratio.*

Proof. We denote by $C(0, N)$, the complexity of solving an SSG with 0 MAX vertices and $N = |V|$ total vertices. This is the resolution of a one-player game, and can be done in polynomial time in the size of the game by solving a linear programming problem. We define $C(n, N)$ as:

$$C(n, N) = C(n-2, N) + C(n-3, N) + \ldots + C(1, N) + 3C(0, N)$$

We show by induction that the complexity of solving an SSG using Algorithm 8 with n MAX vertices and k total vertices is bounded by $C(n, N)$. According to Proposition 5.14, each call to DecreasingFixedSet is done on an SSG with a decreasing number of MAX vertices. Proposition 5.13 also stipulates that if S_i is not empty, then $S_{i-1} \setminus S_i$ is also not empty and $|T_i|$ is greater than one. Thus, each recursive call is made on an instance with at most $n-2$ MAX vertices. Finally, v_σ is computed twice before the loop, costing $C(0, N)$ operations. Therefore, Algorithm 8 has time complexity $O(C(n, N))$. We notice that $C(n, N) - C(n-1, N) = C(n-2, N)$. Thus, we have:

$$C(n, N) = O(\varphi^n Poly(|G|))$$

Thus, we have shown that Algorithm 8 has time complexity $O(\varphi^n Poly(|G|))$. □

Although Algorithm 8 can be extended to SSG of degree d, the complexity analysis does not hold in the general case. Indeed, T does not strictly decrease at each step of the algorithm.

The polynomial factor in all our Algorithms corresponds to the complexity of computing v_σ from σ. We recall that this is the complexity of solving a linear programming problem with $|V|$ variables. It is the same polynomial factor as the one in Tripathi, Valkanova and Kumar's algorithm [TVK11] which runs in $O\left(2/n \cdot Poly(|G|)\right)$.

However, the analysis of Algorithm 8 does not hold in the case of SSG with higher degree. Algorithm 7 can still be improved for some degree by changing the size of the fixed set according to d. For instance, if we fix a set of size 3, the complexity of solving SSG of degree 3 is $O\left(17^{n/3}\right)$ iterations instead of $O\left(7^{n/2}\right)$. For information, $17^{1/3} \simeq 2.57$ and $7^{1/2} \simeq 2.65$. However, increasing the size of the fixed set not always yield better complexity. For binary SSG, the number of iterations with set of size 2 is $O\left(3^{n/2}\right)$ and $O\left(6^{n/3}\right)$ for set of size 3, and we know that $6^{1/3} \simeq 1.82$ and $3^{1/2} \simeq 1.73$.

6 - SSG as a Unique Sink Orientation Problem

Contents

The functional problem of finding an optimal MAX strategy of an SSG can be reduced to a unique sink orientation (USO) problem on cubes for SSG of degree 2 and on grids for SSG of arbitrary degree [BSV03, BSV04, GJMR08]. We start by presenting the USO problem, then we give a simple proof of why SSG can be seen as a USO problem, and finally we present some known algorithms for solving USO. While it is known in the USO community that solving an SSG can be interpreted as a USO problem, this fact is not well-known in the SSG community, as shown by the number of papers presenting the Hoffman-Karp algorithm with a bound of $O(2^n/n)$ iterations as the best deterministic algorithm parametrised by n.

6.1 . The Unique Sink Orientation Problem

6.1.1 . The Unique Sink Orientation on Polytope

Usually, a polytope can be defined as an intersection of half-spaces, as it is the case in linear programming or the convex hull of vertices in $\mathbb{R}^n$. The faces of a polytope $\mathcal{P}$ are the sets F such that there exist a hyperplane H that supports $\mathcal{P}$ and such that $\mathcal{P} \cap F = \emptyset$. For more information on polytope, we refer to [G+03]. The geometric background behind polytope is not relevant to study the unique sink orientation problem. We thus give the following definition of polytope.

Definition 6.1 (Polytope and Faces)**.** *A polytope is defined as an undirected connected graph, and the faces of the polytope are a family of induced subgraphs containing the original graph and all subgraphs containing exactly one vertex.*

For instance, a triangle, the complete graph on three vertices, is a polytope, and we can consider as faces every induced subgraphs of it. In this thesis, we only consider three families of polytope that we will precisely define: cubes and grids in this chapter and permutahedrons in Chapter 7.

Definition 6.2 (Unique Sink Orientation Property)**.** *Let $\mathcal{P}$ be a polytope. A unique sink orientation is an orientation of the edges such that for each face of the polytope, there is exactly one vertex with out-degree zero, called the sink of the face.*

Definition 6.3 (Orientation Oracle)**.** *Given a polytope and a unique sink orientation of this polytope, an orientation oracle is an oracle that, given a vertex of the polytope, returns the orientation of its adjacent edges.*

Definition 6.4 (The USO Problem)**.** *The Unique Sink Orientation Problem is, given an orientation oracle of a polytope, the functional problem of finding the sink of the polytope.*

Remark 6.5. When solving a USO problem, we require calling the oracle on the sink, even if we already know it is the sink. It is justified, because often the oracle gives another information that is of interest to us. For instance, we later show that for the USO corresponding to an SSG, we compute the value vector of the strategy associated to each evaluated vertex. Another reason is while considering recursive algorithms on faces of the polytope, which gives an orientation of the edges connected to the sink and not in the face.

We now define the hypercube, or cube, the grid and their faces.

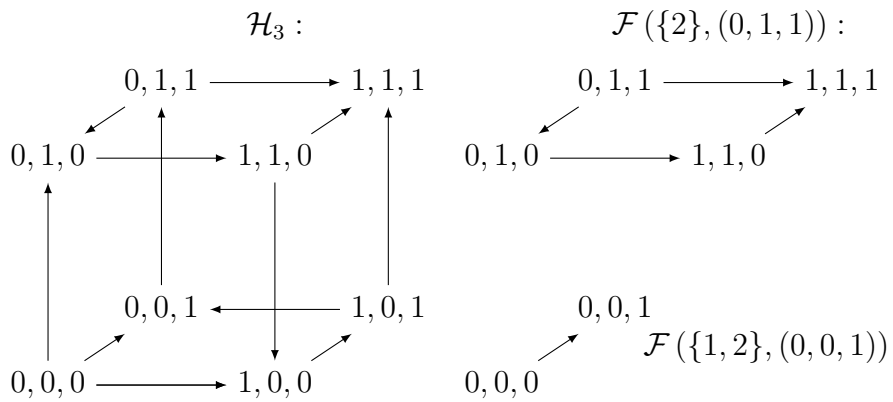


Figure 6.1: Representation of $\mathcal{H}_3$ and two faces of dimension 2 and 1 satisfying the unique sink orientation property

Definition 6.6 (Hypercube)**.** *The* **hypercube** $\mathcal{H}_n$ *is the graph with vertex* V *and edge* E *with* $V = \{0,1\}^n$*:*

$$E = \{(x,y) \mid \exists i, x_i \neq y_i \text{ and } \forall j \neq i, x_j = y_j\}$$

The faces of a hypercube are the hypercubes of lower dimension contained in it.

Definition 6.7 (Face of a Hypercube)**.** *For* $\mathcal{H}$ *a hypercube,* x *a vertex of* $\mathcal{H}$ *and* $I \subseteq \{1,\dots,n\}$*, the face* $\mathcal{F}(I,x)$ *is the subgraph induced by the set of vertices* $V(I,x)$ *defined as:*

$$V(I,x) = \{y \in V \mid \forall i \in I, x_i = y_i\}$$

We present an oriented cube and two of its faces in Figure 6.1. One can notice that the orientation of the cube satisfies the unique sink orientation property. However, it is not acyclic: the acyclicity condition is not necessary for solving USO and is not required for any of the algorithm presented in this chapter.

We now define grids as presented in [BMN+19]. First, we recall the definition of the Cartesian product of two graphs.

Definition 6.8 (Cartesian Product)**. The Cartesian product** *of two graphs* $G = (V,E)$ *and* $G' = (V',E')$ *denoted by* $G \square G'$ *is the graph* $H = (V_H, E_H)$ *with:*

$$V_H = V \times V'$$
$$E_H = \{\big((u,v),(u',v)\big) \mid (u,u') \in E\} \cup \{\big((u,v),(u,v')\big) \mid (v,v') \in E'\}$$

Definition 6.9 (Grid)**.** *Let* $n \geq 1$ *and* $d_1,\dots,d_n$ *be integers greater or equal than* 2*. The Cartesian product* $K(d_1)\square\dots\square K(d_n)$ *where* $K(d)$ *is the complete graph with* d *vertices is called a* **grid of dimension** n *.*

It is quite clear that a cube of dimension n is a grid that is a product of n graphs $K(2)$. We authorise the grid to have some components with only 1 vertex, and in this case we diminish the dimension accordingly. For instance, if we consider the grid $K(5)\square K(1)\square K(2)\square K(1)$, we notice that it is isomorphic to the grid $K(5)\square K(2)$ and thus can be considered to be a grid of dimension 2.

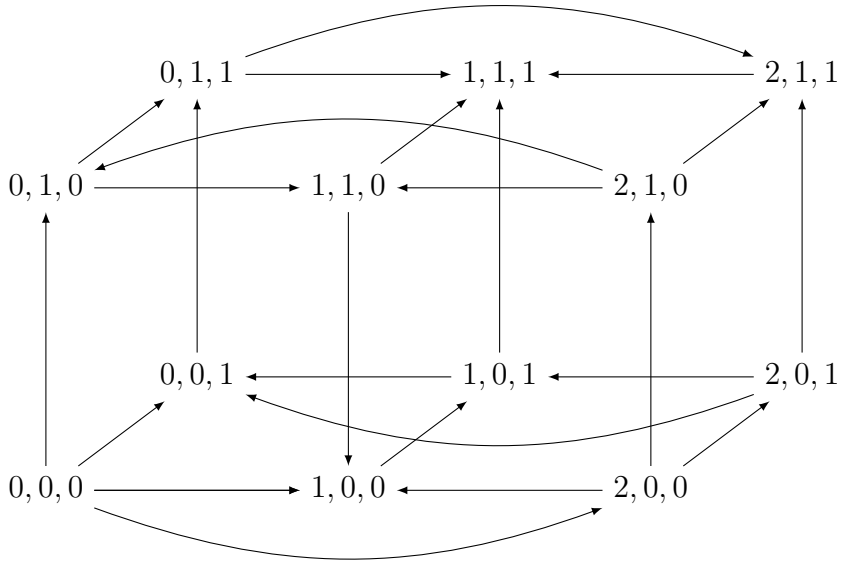


Figure 6.2: A unique sink orientation of the grid $K(3)\square K(2)\square K(2)$

Remark 6.10. For consistency, we use the notation n for the dimension of the grid and d for the degree because in the reduction from SSG to USO, the dimension of the grid corresponds to the number of MAX vertices. Notice that in some paper on USO, for instance [BMN+19], the two notations are switched.

The faces of the grid are also called subgrids.

Definition 6.11 (Subgrid)**.** *For all $i \leq n$ and $I_i \subseteq \{1, \ldots, d_i\}$, the graph induced by the Cartesian product $I_1 \times \ldots \times I_n$ is a face of the grid, also called a subgrid.*

We notice that a subgrid is also a grid. Moreover, the grid is a subgrid of itself. We give an instance of a grid in Figure 6.2.

6.1.2 . SSG as a USO Problem

In order to show that the functional SSG problem can be reduced to the USO problem, we show that a stopping SSG can be modelled as a USO. We recall that a general SSG can be approximated by a stopping SSG by adding a quadratic number of random vertices (Corollary 2.6). We show at the end of this section an instance of non-stopping SSG for which the reduction is not immediate.

Let G be a stopping SSG with n MAX vertices with degree $d_1, \ldots, d_n$. We denote by $x_1, \ldots, x_n$ the MAX vertices of G and for any MAX vertex x_i, we write its out-neighbourhood as $x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(d_i)}$. The MAX positional strategies are in bijection with the elements of $\{(k_1, \ldots, k_n) \mid \forall i, 1 \leq k_i \leq d_i\}$. We consider an arbitrary order on the set of all vertices.

We consider the grid $Gr = K(d_1)\square \ldots \square K(d_n)$. To each vertex $(k_1, \ldots, k_n)$ of Gr, we associate the strategy σ such that $\sigma(x_i) = x_i^{(k_i)}$ for every $i \leq n$. The edges of the grid thus correspond to pairs of strategies that differ on only one vertex.

We consider the following orientation $\mathcal{O}$ defined for the vertex $(k_1, \ldots, k_n)$ corresponding to strategy σ and $(k_1, \ldots, k_{i-1}, k'_i, k_{i+1}, \ldots, k_n)$ corresponding to strategy σ'.

$$\mathcal{O}(\sigma, \sigma') = \begin{cases} \sigma' \text{ if } k'_i \in IS_\sigma(x_i) \\ \sigma' \text{ if } v_\sigma(x^{(k_i)}) = v_\sigma(x^{(k'_i)}) \text{ and } x^{k'_i} < x^{k_i} \\ \sigma \text{ otherwise} \end{cases}$$

We recall that the Improvement Set denoted by $IS_\sigma(x)$, presented in Definition 3.15, is the set of vertices towards which a switch is profitable. The second case in our definition happens when two strategies have the same value vector. In that case, we follow the previously defined arbitrary order on the vertices. Thus, computing the orientation of a vertex can be done by computing the value vector of the strategy, which can be done in polynomial time by Proposition 1.19. Thus, the cost of calling the orientation oracle is the cost of solving a one player game.

It remains to prove that the grid Gr together with the orientation $\mathcal{O}$ satisfies the unique sink orientation property. Since the degree of a coordinate of Gr model the degree of the associated MAX vertex, reducing the degree by considering a subgrid is equivalent to remove the corresponding edges in the game. It results that the subgrids of Gr are subgames of G where some edges have been removed. Hence, it is enough to prove that for every game G and associated grid and orientation Gr and $\mathcal{O}$, there is a unique sink.

Lemma 6.12. *Let G be an SSG and Gr and $\mathcal{O}$ be the grid and orientation created using above method. Then, Gr has a unique sink.*

Proof. First, we show that Gr has at least one sink. There is at least one positional MAX strategy σ^* such that $v_{\sigma^*} \geq v_\sigma$ for all strategies σ by Proposition 1.27. Let M be the set of all such strategies. Every outgoing edge from a vertex of M is towards a vertex of M smaller following the lexicographic order extending the chosen order on the set of vertices. Hence, there is at least one sink of Gr in M.

Now, for the sake of contradiction, let us assume that there are two sinks σ and σ'. By definition, their switch sets are empty, thus they are both optimal strategies with the same value vector. Let i be the smallest index such that $\sigma(x_i) = x_i^{(k_i)} \neq \sigma'(x_i) = x_i^{(k'_i)}$. W.l.o.g. we assume that $x^{k_i} < x^{k'_i}$ following the order on vertices defined above, and we consider the strategy σ'' that plays as σ' except in x_i where $\sigma''(x_i) = x^{(k_i)}$. Since G is a stopping game, and σ and σ' are optimal strategies, then for $\tau(\sigma)$ a best response to σ, $(\sigma', \tau(\sigma))$ satisfies the optimality condition of Lemma 1.22 and $v_{\sigma'} = v_{\sigma''}$ and σ'' is smaller under the lexicographic order than σ'. Hence, σ' is not a sink, and we reached a contradiction. □

Remark 6.13. This orientation is not enough to obtain the unique sink orientation property for general SSG. Indeed, we consider the SSG with two MAX vertices x_1, x_2 and one sink 1 and we consider the following edges:

$$x_1^{(1)} = x_2,\ x_1^{(2)} = 1,\ x_2^{(1)} = x_1,\ x_2^{(2)} = 1$$

and the following order on vertices $x_1 < x_2 < 1$In this case, we obtain two sinks in $(1, 2)$ and $(2, 1)$. We present this case in Figure 6.3. This problem is solved by considering stopping SSG.

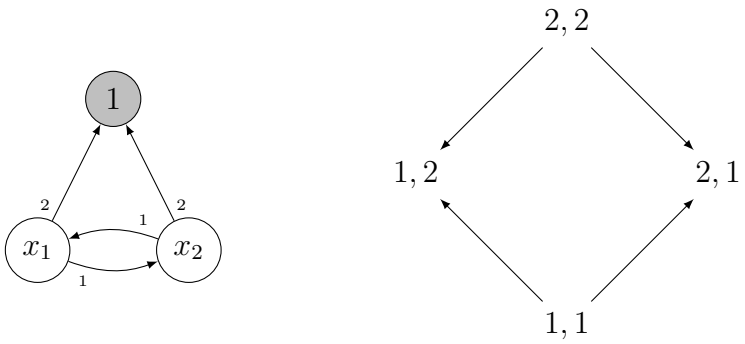


Figure 6.3: Non-stopping SSG cannot easily be represented as a USO problem

6.2 . The USO Problem on Cubes

6.2.1 . The USO Problem on Cubes

The unique sink orientation problem restricted to hypercubes has been introduced by Stickney and Watson in [SW78] for studying linear complementarity problems and further studied by Szabó and Welzl under the name unique sink orientation in [SW01]. We provide different notations than the one of the original article to be closer to the notations used for SSGs.

For $\mathcal{F}$ a face of $\mathcal{H}_n$, we write $s(\mathcal{F})$ the sink of $\mathcal{F}$. We also write $\mathcal{F}(I, x)$ for x a vertex of $\mathcal{H}_n$ to represent the face formed by the vertices that have same value as x on coordinates in I. Let x be a vertex of $\mathcal{H}_n$, we denote by $\overline{x}^k$ the vertex y such that $x_i = y_i$ for all $i \neq k$ and $x_k \neq y_k$.

An orientation of $\mathcal{H}_n$ is a function $\mathcal{O}$ from E to V which for each edge of E choose one of its two endpoints. For I a subset of $\{1, \ldots, n\}$, $\overline{\mathcal{O}}^I$ is the orientation defined as:

$$\forall i \notin I,\ \forall x \in V, \overline{\mathcal{O}}^I((x, \overline{x}^i)) = \mathcal{O}((x, \overline{x}^i))$$

$$\forall i \in I,\ \forall x \in V, \overline{\mathcal{O}}^I((x, \overline{x}^i)) \neq \mathcal{O}((x, \overline{x}^i))$$

Informally, $\overline{\mathcal{O}}^I$ corresponds to the orientation where all the edges on the coordinates of I have been changed.

Lemma 6.14 ([SW01]). *If $\mathcal{O}$ is a unique sink orientation, then for every $I \subset \{1, \ldots, n\}$, $\overline{\mathcal{O}}^I$ is a unique sink orientation*

For a given orientation, we can define the switch set of a vertex. In [SW01], this is called an out-map.

Definition 6.15 (Switch set). *For a hypercube $\mathcal{H}$ and an orientation $\mathcal{O}$, the switch set of a vertex x is defined as:*

$$S^{\mathcal{H}}_{\mathcal{O}}(x) = \{i \in \{1, \ldots, n\} \mid \mathcal{O}(x, \overline{x}^i) = \overline{x}^i\}$$

if there is no confusion, we omit $\mathcal{H}$ and $\mathcal{O}$.

The switch set of a vertex corresponds to the set of coordinates for which the associated edges are outgoing. Hence, we are searching x such that $|S_{\mathcal{O}}(x)| = 0$. We justify using the name of switch set thanks to the following lemma that remind us of Proposition 5.3.

Lemma 6.16 ([SW01]). *There is a bijection between vertices and their switch set.*

Proof. We provide the proof of [SW01]. If under orientation $\mathcal{O}$ there is u and v such that $S(u) = S(v)$, then we consider the orientation $\overline{\mathcal{O}}^{S(u)}$ in which both u and v are sinks. Hence, by Lemma 6.14, $u = v$. Thus, the switch set is injective and by cardinality, there is a bijection. □

One of the main interest of abstracting a problem as an USO problem on cube is that it is possible to take the quotient of the cube by some of its faces.

Definition 6.17 (Inherited cubes). *For $\mathcal{H}$ a cube of dimension n and $I \subseteq \{1, \dots, n\}$, the inherited cube $\mathcal{H}^I$ is the cube of dimension $|I|$.*

While considering $\mathcal{H}^I$, the coordinates of the vertices of the inherited cubes are the $(x_{i_1}, \dots, x_{i_I})$ with for all k, $i_k \in I$. In other words, it is the cube where the coordinates not in I have been removed.

Definition 6.18 (Inherited orientation). *The inherited orientation $\mathcal{O}^I$ of $\mathcal{H}^I$ is defined as*

$$\forall i \in I,\ \mathcal{O}^I(x, \overline{x}^i) = \overline{x}^i \text{ if and only if } \mathcal{O}\left(s(\mathcal{F}(I,x)), \overline{s(\mathcal{F}(I,x))}^i\right) = \overline{s(\mathcal{F}(I,x))}^i$$

In other words, the orientation of the edges of $\mathcal{O}^I$ correspond to the orientations of the edges in the neighbourhood of the sinks of the faces $\mathcal{F}(I,x)$. We present the example of [SW01] in Figure 6.4. If we translate this to the context of SSG, a face corresponds to the game where some edges have been fixed and thus a sink of a face correspond to a super-switch as defined in Chapter 5. Hence, the inherited cubes of an SSG corresponds to the study of super-switches.

Lemma 6.19 ([SW01]). *The Inherited orientation $\mathcal{O}^I$ of $\mathcal{H}^I$ defines a unique sink orientation.*

Proposition 6.20 ([SW01]). *Let $t(n)$ the maximal number of calls to the oracle needed to solve the USO problem on a hypercube of dimension n, then for all $1 \leq k < n$:*

$$t(n) \leq t(n-k) \cdot t(k).$$

Proof. Let $I \subseteq \{1, \dots, n\}$ of dimension k and consider the cube $\mathcal{H}^I$ with orientation $\mathcal{O}^I$. Finding the sink of $\mathcal{H}$ can be done by computing the sink of the face corresponding to the sink of $\mathcal{H}^I$. Finding the sink of $\mathcal{H}^I$ can be done with $t(k)$ evaluation of its vertices. Evaluating a vertex correspond to solving the problem USO on a face of $\mathcal{H}^I$, hence, it corresponds to solving a USO problem on a cube of dimension $n-k$. Hence, we can find the sink of $\mathcal{H}$ in $t(n-k) \cdot t(k)$ calls to the oracle. □

It is easy to show that $t(2) = 3$. Indeed, one can evaluate a vertex and the vertex opposite to it, thus determining the orientation of each edge and thus finding the sink. We recall that we still need to call the oracle on the sink, thus $t(2) \leq 3$. By brute-forcing every possibility, one can see that $t(2) = 3$. There are only two possible orientations in dimension 2, as shown in Figure 6.5. Thus, using a recursive argument and Proposition 6.20, there is an algorithm using $t(2)^{\lceil n/2 \rceil}$ call to the oracle that solves USO on cubes. Hence, an algorithm solving recursively faces of dimension 2 in 3 evaluations uses at most $3^{\lceil n/2 \rceil}$ calls to the oracle. This proves the correctness of Algorithm 7 for SSG of degree 2 and proves that it also works to solve USO on cubes.

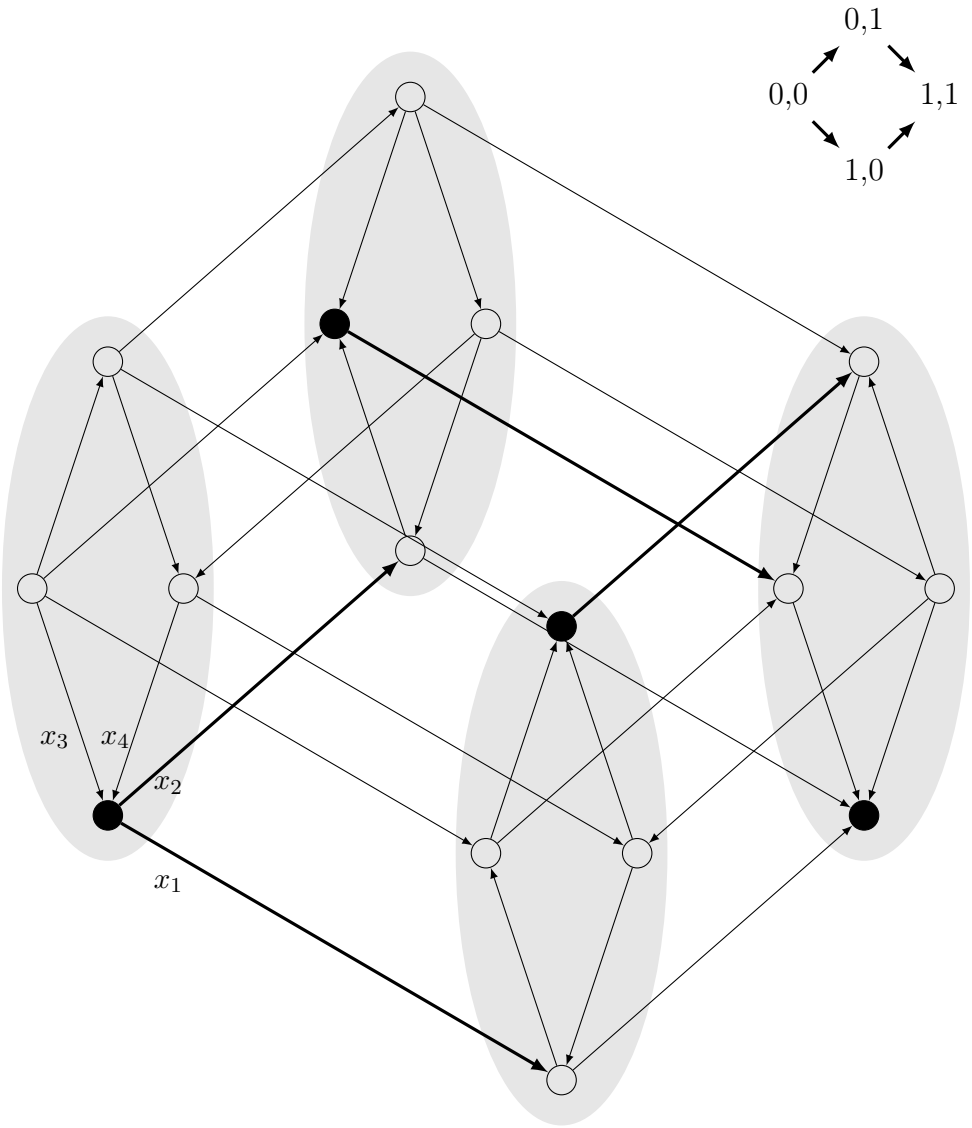


Figure 6.4: The Instance of the inherited cube $\mathcal{H}^{\{3,4\}}$ presented in [SW01].

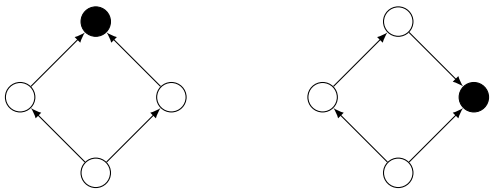


Figure 6.5: The two possible configurations for orientation on $\mathcal{H}_2$

6.2.2 . The Fibonacci Seesaw Algorithm

In order to present the Fibonacci Seesaw Algorithm of [SW01], let us define antipodal faces.

Definition 6.21 (Antipodal Faces)**.** *Let $I \subseteq \{1, \dots, n\}$ and x and x' be two vertices such that $x_i \neq x'_i$ for all $i \in I$. Then, the faces $\mathcal{F}(I, x)$ and $\mathcal{F}(I, x')$ are said to be* **antipodal faces**.

In other words, antipodal faces are faces where the fixed vertices are all different. The concept of the Fibonacci Seesaw Algorithm is to simultaneously compute the sinks of two antipodal faces of the hypercube.

Let $F = \mathcal{F}(I, x)$ and $G = \mathcal{F}(I, y)$ be two antipodal faces of $\mathcal{H}$ of dimension $n - |I| = k$ with respective sinks u and v already evaluated. We want to compute the sink of two antipodal faces of dimension $k + 1$. We know that $S^{\mathcal{H}}_{\mathcal{O}}(u) \neq S^{\mathcal{H}}_{\mathcal{O}}(v)$ by Lemma 6.16. By symmetry, we suppose that there is i in $S^{\mathcal{H}}_{\mathcal{O}}(u) \setminus S^{\mathcal{H}}_{\mathcal{O}}(v)$. Moreover, since u is the sink of F, i is in I. If $k = n - 1$, then v is a sink of $\mathcal{H}$. Otherwise, we consider the antipodal faces $F' = \mathcal{F}(I \setminus \{i\}, u)$ and $G' = \mathcal{F}(I \setminus \{i\}, v)$. We have $F \subsetneq F'$ and $G \subsetneq G'$. Moreover, v is the sink of G'. We need to compute the sink of F'. We know that the sink of F' is not u and thus is not in F. Therefore, the sink of F' is in $\mathcal{F}(I, \overline{u}^i)$. Hence, it can be found by computing a sink in a cube of dimension k.

The Algorithm starts by computing the orientation of two antipodal vertices which corresponds to the sink of antipodal faces of dimension 0 then, uses the above method to increase the dimension of the two antipodal faces until finding the sink of $\mathcal{H}$.

Theorem 6.22. *The Fibonacci Seesaw Algorithm needs $O(\varphi^n)$ calls to the orientation oracle, with $\varphi = \dfrac{1+\sqrt{5}}{2}$ the golden ratio.*

Proof. Let $t'(n)$ be the number of call to the oracle necessary to compute the sink of a hypercube of dimension n using this algorithm. We start by calling the oracle twice, then we have:

$$t'(n) \leq 2 + t'(0) + t'(1) + \ldots + t'(n-2)$$

with $t'(0) = 1$. Hence, $t'(n) = O(\varphi^n)$. □

Remark 6.23. In the case of SSG, we notice that the Fibonacci Seesaw algorithm informally corresponds to some worst case situation of Algorithm 8. Indeed, switching all vertices correspond to computing the sink of the antipodal faces. In other words, in Algorithm 8, if the switch set of the current strategy is of maximal size, so is reduced by one at each iteration, the next considered strategy is the sink of the antipodal face.

This bound can still be improved. Indeed, in their paper, Szabó and Welzl [SW01] also found a method to solve USO on cubes of dimension 4 with at most 7 calls to the oracle while the Fibonacci Seesaw algorithm needs 8. With this transformation, the number of calls to the oracle $C(n)$ is defined by $C(0) = 1$, $C(1) = 2$, $C(2) = 3$, $C(3) = 5$ and for $n \geq 4$:

$$C(n) \leq 2 + \sum_{i=0}^{n-5} C(i) + 5C(n-4).$$

The $5C(n-4)$ comes from the fact that two sinks of faces of dimension $n-4$ are found by Fibonacci Seesaw, and they can be used to find the global sink with five more resolutions on faces of dimension $n-4$. This result in a complexity of $O(1.61^n) = o(\varphi^n)$

6.3 . Algorithm for Solving USO on Grids

6.3.1 . Properties of Grids Orientation

We now consider the USO problem on grids. To do so, we mostly use the notations used in [BMN+19]. An extension to inherited orientations on cubes called induced orientations can be defined for grids.

Definition 6.24 (Induced Grid)**.** *Let $\mathcal{P} = (\mathcal{P}_i)_{1\leq i\leq n}$ with, for all i, $\mathcal{P}_i$ a partition of the vertices of $K(d_i)$. The inherited grid $G(\mathcal{P})$ is defined as:*

$$G(\mathcal{P}) = K(\mathcal{P}_1)\square\ldots\square K(\mathcal{P}_n).$$

In order to define the orientation of $G(\mathcal{P})$, we start by noticing that for $u = (u_1, \ldots, u_n)$ a vertex of $G(\mathcal{P})$, u can be identified as a subgrid of G

$$\overline{u} = K(u_1)\square\ldots\square K(u_n).$$

We denote by $sink(\overline{u})$ the sink of the face $\overline{u}$. For two adjacent vertices u and v, $\overline{u}\cup\overline{v}$ is also a subgrid of G. We can now define the induced orientation.

Definition 6.25 (Induced Orientation)**.** *The induced orientation $\mathcal{O}^{\mathcal{P}}$ of $G(\mathcal{P})$ is defined for u and v adjacent vertices as:*

$$\mathcal{O}^{\mathcal{P}}(u,v) = \begin{cases} v \text{ if } sink(\overline{u}\cup\overline{v}) = sink(\overline{v}) \\ u \text{ otherwise} \end{cases}$$

We present an instance of Induced Orientation in Figure 6.6. As for inherited orientation, the orientation of the induced grid is defined by the orientation of the sink of the faces represented by each vertex. A restricted version of induced orientation called inherited orientation is presented in [GJMR08] where each partition is either the entire set or a partition into singletons. In this situation, either the coordinate is fixed or is totally free.

Proposition 6.26 ([BMN+19])**.** *If s is a sink in $G(\mathcal{P})$ then $sink(\overline{s})$ is the sink of G.*

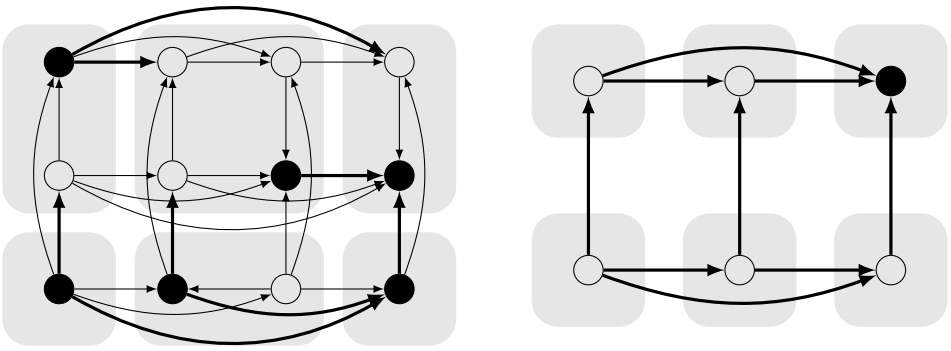


Figure 6.6: The inherited grid of a grid $K(4)\square K(3)$.

This implies that $\mathcal{O}^{\mathcal{P}}$ is a unique sink orientation.

In terms of SSG, the vertices $u = (u_1, \ldots, u_n)$ of an induced grid correspond to the optimal MAX strategy in the game where all edges not in u_i have been removed.

Similarly to SSG, we can define a switch vector for grids, as shown in [GJMR08].

Definition 6.27 (Improvement Set)**.** *Let G be a grid and $\mathcal{O}$ a unique sink orientation of G. Let u be a vertex of G. The Improvement Set of u written IS_u is defined as:*

$$IS_u(i) = \{k \in K(d_i) \mid \exists v, \forall j \neq i, u_j = v_j, v_i = k, \text{ and } \mathcal{O}(u, v) = v\}$$

Definition 6.28 (Switch vector)**.** *The switch vector $\vec{S}_v$ is defined as*

$$\vec{S}_v(i) = |IS_u(i)|$$

Proposition 6.29 ([GJMR08])**.** *The switch vector is a bijection between V and $\{0, \ldots, d_1 - 1\} \times \ldots \times \{0, \ldots, d_n - 1\}$.*

In [GJMR08], the switch vector is called the refined index of $\mathcal{O}$.

6.3.2 . Algorithm for Solving USO on Grids

In this section, we present the Algorithm of [BMN+19]. The unique sink orientation modelling Simple Stochastic Games are acyclic, however, the algorithms that we present in this chapter works for general USO. To do so, we start by focusing on grids of dimension 2.

Definition 6.30 (Rows and Columns)**.** *In a grid of dimension 2, the row correspond to the first coordinate and the column to the second. The i^{th} row is the set of vertices (i, j) for all j.*

In [GJMR08], they proved an acyclicity result on USO of grids of dimension 2.

Lemma 6.31 ([GJMR08])**.** *USO on grids of dimension 2 are always acyclic.*

This result and Proposition 6.29 are enough to see that Algorithm 7 presented in Chapter 5 on SSGs also works on grids, using the same number of iterations.

Lemma 6.32 (Row and Column Elimination [BMN+19])**.** *Let $G = K(m) \square K(m)$ be a grid of dimension 2 and $\mathcal{O}$ a unique sink orientation of G. We suppose that we have evaluated m vertices all in different rows and columns. It is possible to compute in quadratic time, with no additional call to the oracle, indices i_0 and j_0 such that the sink of G is neither in row i_0 or column j_0.*

Proof. The idea of the proof is to notice that, using acyclicity of grids of dimension 2 (Lemma 6.31) it is possible to renumber the evaluated vertices such that for any subgrid of degree 2 containing $v_m = (m, m)$, v_m has in-degree at most 1. This implies that for I and J defined as:

$$I = \{i < m \mid \mathcal{O}((i, m), v_m) = v_m\}$$

$$J = \{j < m \mid \mathcal{O}((m, j), v_m) = v_m\}$$

we have $I \cup J = \{1, \ldots, m-1\}$. In the subgrid $G' = K(I \cup \{m\}) \square K(J \cup \{m\})$ we notice that v_m is the sink of G'. Hence, the sink of G is not in $I \times J$.

We compute recursively a row i_0 in which the sink is not located in $I \times I$ and a column j_0 in which the sink is not located in $J \times J$. Hence, for all $k \leq m$, the sink of G is not (i_0, k) or (k, j_0). □

If there are more columns than rows, a similar method can be used to eliminate a column. In this case, for I the set of rows and J the set of columns, we consider the subgrid defined by $I \times J'$ with $|J'| = |I|$ and we apply Lemma 6.32 which gives a column j_0 in which the sink of $I \times J'$ does not appear. Hence, it does not appear in column j_0 in the grid $I \times J$.

Corollary 6.33 (Column Elimination [BMN+19]). *Let $G = K(m)\square K(m')$ be a grid of dimension 2 with $m \leq m'$ and $\mathcal{O}$ a unique sink orientation of G. We suppose that we have evaluated m vertices all in different rows and columns. It is possible to compute in quadratic time, with no additional call to the oracle, an index j_0 such that the sink of G is not in column j_0.*

This gives Algorithm 9 for grids of degree 2.

Algorithm 9: RecursivePair [BMN+19]

Data: $G = K(d_1)\square K(d_2)$ a grid with unique sink orientation $\mathcal{O}$
Result: The sink of G

```
1  begin
2      We assume that d_1 ≤ d_2
3      Evaluate vertices (1,1), ..., (d_1, d_1)
4      while d_1 < d_2 do
5          Use Corollary 6.33 to find a column j_0 that does not contain the sink
6          Delete j_0 of G
7          Evaluate at most another vertices such that d_1 vertices with different rows and
             columns of G are evaluated
8      while d_1 ≥ 2 do
9          Use Lemma 6.32 to find a row i_0 and column j_0 that does not contain the sink
10         Delete i_0 and j_0 of G
11         Evaluate at most another vertices such that d_1 vertices with different rows and
             columns of G are evaluated
12     return The coordinates of the remaining vertices
```

At the line 7 and 11 of Algorithm 9 is always possible to maintain having d_1 vertices of G evaluated all with different rows and columns by evaluating at most another vertices at each iteration. Indeed, if for instance row i_0 and j_0 are eliminated and if vertices (i_0, j) and (i, j_0) were already evaluated, by evaluating (i, j) we maintain our invariant. If only (i_0, j_0) was evaluated, no further evaluation would be required. This gives the following complexity result.

Proposition 6.34 ([BMN+19]). *Algorithm 9 requires at most $d_1 + d_2 - 1$ calls to the oracle to solve the USO problem on $G = K(d_1)\square K(d_2)$.*

This bound is tight for grids of dimension 2 [BMN+19]. We recall that USO of larger dimension can be solved recursively. Hence, if we only consider the first two coordinates in Algorithm 9, evaluating a vertex corresponds to solving a USO problem of dimension $n - 2$.

Theorem 6.35 ([BMN+19]). *The sink of a unique sink orientation problem on grids of dimension n and degree d can be found in $O\left(d^{\lceil n/2 \rceil}\right)$ calls to the orientation oracle.*

6.4 . Perspective for SSG

The two algorithms presented in this chapter are the best deterministic algorithms for solving SSGs. The fact that the best deterministic algorithms on SSG come from USO and do not use some properties specific to SSG, could imply the existence of faster algorithms. Some important study points contain acyclicity, the study of randomised strategies that are interior points, and the existence of a second player. Indeed, even on simpler games, like parity game or Markov decision process, strategy improvement does not yield any sub-exponential algorithm.

An other interesting point to notice is that Algorithm 9 is heavily reliant on the fact that Grids of dimension 2 are acyclic. This raises the question of whether the acyclicity criterion could be further leveraged in similar algorithms.

7 - Seeing SSG as a USO Problem on the Permutahedron

Contents

Since modelling an SSG by a USO on grids gives good algorithms, we try to represent SSG as a USO problem on a different polytope. In this chapter, we model the functional problem for SSG as a USO problem on the permutahedron, a polytope in dimension $n-1$ whose set of vertices is the symmetric group of degree n $\mathcal{S}_n$: the set of permutations of $\{1,\dots,n\}$.

7.1 . Presentation of the Permutahedron

Once again, geometric considerations on the permutahedron are not relevant to us, hence we only present it as a graph. For more details, one can for instance consult [Tho06]. In this chapter, we write $\mathcal{S}_n$ for the symmetric group: the set of permutations of the n first integers. We use the notation f for an element of $\mathcal{S}_n$, for consistency with Section 4.3.

Definition 7.1 (Permutahedron of order n)**.** *The Permutahedron of order n is the graph $\Pi_n = (S_n, E)$ with:*

$$E = \{(f, f') \mid \exists i < n, f \circ (i\ i+1) = f'\}$$

We give a geometric representation of Π_4 in Figure 7.1 and a representation of the graph in Figure 7.2. The vertices of Π_n are the set of **total orderings** of $\{1,\dots,n\}$, and two orderings are adjacent if and only if they *differ on only two elements*. Notice that this is coherent with the original definition of permutahedron as coined by Guilbaud and Rosenstiehl in [GR63]. This graph is the Cayley graph of the symmetry graph generated by the transpositions that swap consecutive elements. For more details on Cayley graphs, one can for instance look at [GR01].

Definition 7.2 (Faces of the Permutahedron)**.** *Let f be an element of Π_n and $I \subseteq \{1,\dots,n-1\}$. We consider the graph Π'_n where all edges $(f, f \circ (f(i), f(i+1)))$ have been removed for $i \in I$. The connected component $\mathcal{F}(I, f)$ containing f is a face of Π_n.*

In other words, the face $\mathcal{F}(I, f)$ is the Cayley graph generated from the permutation f by the permutations that swap consecutive elements in position i and $i+1$ for $i \notin I$. We can now define a unique sink orientation problem on the permutahedron. Given an orientation $\mathcal{O}$ of the edges of the permutahedron, such that every face has exactly one sink, the associated functional problem is to find the sink of Π_n.

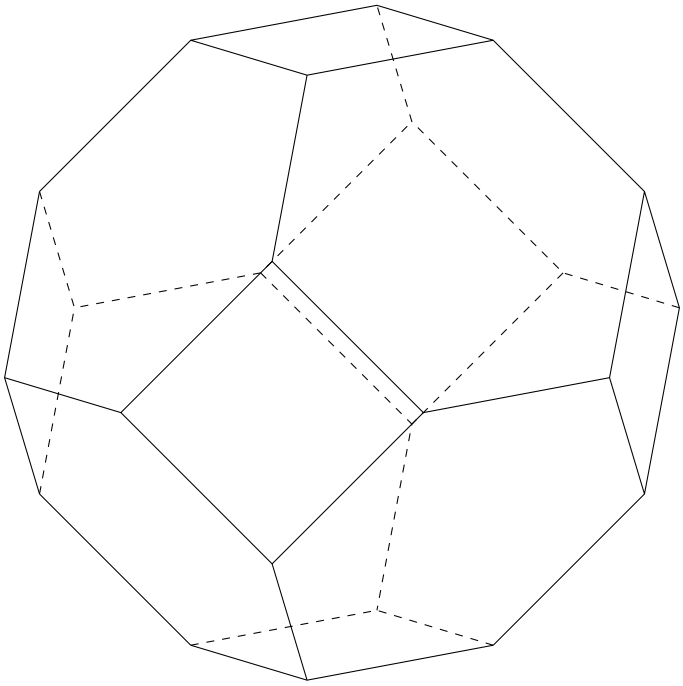


Figure 7.1: Geometric representation of Π_4.

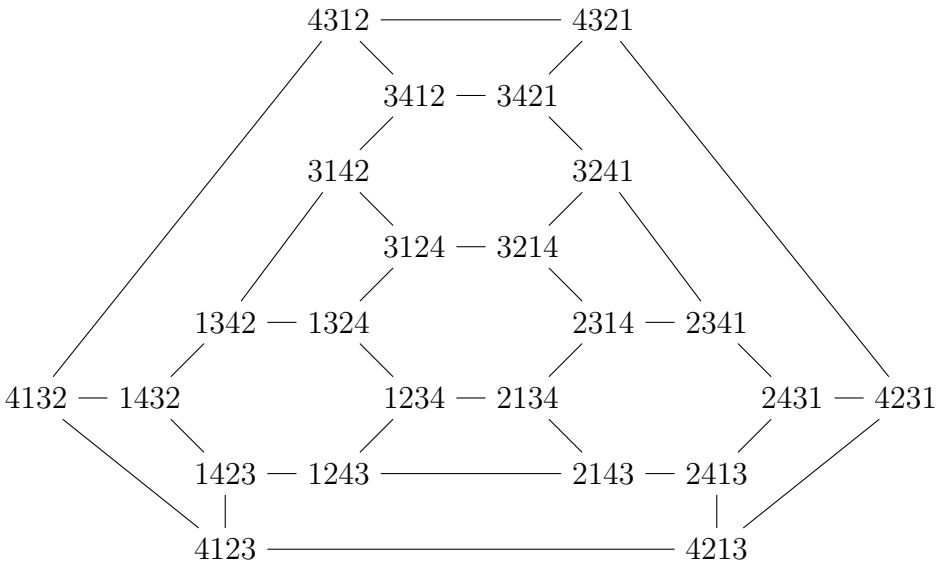


Figure 7.2: Graph of Π_4

7.2 . Expressing SSG as a Problem on the Permutahedron

7.2.1 . Random Stopping SSG

In order to prove that the functional problem of solving SSG can be reduced to a USO problem on the permutahedron, we introduce a variant of the stopping condition that we call the random stopping condition. This is used in [ACS19] for similar reasons, without naming the condition.

Definition 7.3 (Random Stopping Condition)**.** *We say that an SSG satisfies the* **random stopping condition** *if there are two sinks* 0 *and* 1 *and every random nodes of* V_R *has at least one edge towards a sink.*

Considering only 0 and 1 sinks allows us to order the value of random vertices without considering the value of the sinks. As for stopping game in Chapter 2, we define the random ϵ-transformation as the game where an edge towards 0 with associated probability ϵ is added to each random node and the other probabilities are multiplied by $(1-\epsilon)$ in order to be normalise the sum of probability to one.

Lemma 7.4. *For* $\epsilon < q^{-3r}/(2r^2)$*, for any pair of positional strategies, the difference between the value vector of* G *and the random* ϵ*-transformation of* G *are less than* q^{-r}*.*

Proof. The proof is similar to the one of Proposition 2.5 while only considering the Markov chain where MAX and MIN vertices have been removed and replaced by edges between random vertices. □

Notice that this condition does not imply the stopping condition. Moreover, the random ϵ-transformation does not increase the number of random vertices. In [GH08], in order to avoid the stopping condition while considering f-strategies, Gimbert and Horn introduced the notion of **liveness**, informally representing the fact that each random node has an edge towards an attractor of higher value (see Section 4.3 for more details). We notice that for random stopping game, this condition is always satisfied as shown in Lemma 7.5. From now on, in this chapter, all SSG will be considered to be random stopping, without loss of generalities thanks to Lemma 7.4

Lemma 7.5 (Liveness of Random Stopping Games)**.** *Let* σ *be a strategy of* G *a random stopping games, then for every non-null random vertices* a*, there is either an edge towards a vertex of value* 1 *or towards a vector of value greater than* $v_\sigma(a)$*.*

Proof. If there is no edge toward a vertex of value 1, it means that there is a probability ϵ to go to 0. Since $v_\sigma(a) > 0$ it implies that there is x with $(a,x) \in E$ and $v_\sigma(x) > v_\sigma(a)$. □

7.2.2 . Partial Order Game

We recall that we only consider random stopping SSG with two sinks 0 and 1. In order to express the problem of finding an optimal MAX strategy of an SSG as a unique sink orientation problem on the permutahedron, we introduce a subgame similar to the one presented by Auger, Coucheney and Strozecki in [ACS19]. The subgame depends on partial ordering of $\{1,\dots,n\}$. The goal is to associate each face of the permutahedron with the optimal value vector of a subgame.

Definition 7.6 (Ordering)**.** *An ordering* f *of* V_R *is a bijection from* $\{1,\dots,r\}$ *to* V_R*.*

Definition 7.7 (Partial Ordering). *A partial ordering $\tilde{f}$ is a function from $\{1, \dots, k\}$ to the subsets of V_R such that it creates a partition of V_R:*

$$V_R = \tilde{f}(1) \sqcup \dots \sqcup \tilde{f}(k)$$

For $x \in f(k)$ and $y \in f(k')$, with $k < k'$ we write $x <_{\tilde{f}} y$ or say that x is smaller than y according to $\tilde{f}$.

A total ordering of V_R is also a partial ordering with the set of singletons as the partition.

Definition 7.8 (Coherent Partial Order). *Let $\tilde{f}$ and $\tilde{f}'$ be two partial orders. We say that $\tilde{f}'$ is an extension of $\tilde{f}$ if for all x and y such that $x \in \tilde{f}(i)$ and $y \in \tilde{f}(j)$ with $i < j$, then there is $i' < j'$ such that $x \in \tilde{f}'(i')$ and $y \in \tilde{f}'(j')$. We also say that $\tilde{f}'$ is coherent with $\tilde{f}$.*

We can now define the **Partial Order Game**: a subgame of G such that a partial order is forced on the random vertices, using additional MIN vertices as a widget.

Definition 7.9 (Partial Order Game). *Let $G = (V, E)$ be an SSG with r random vertices. Let $\tilde{f}$ be a partial order of V_R with partition $\tilde{f}(1), \dots, \tilde{f}(k)$. The partial order game $G(\tilde{f}) = (V', E')$ is defined as:*

$$V' = V \cup V_R^{\text{MIN}}$$

where V_R^{MIN} is a copy of V_R with every vertex of V_R^{MIN} being a MIN *vertices. For every vertex $x \in V_R$, its copy in V_R^{MIN} is written $\hat{x}$*

$$\begin{aligned} E = &\{(x, y) \in E \mid y \notin V_R\} \cup \{(x, \hat{y}) \mid y \in V_R, (x, y) \in E\} \\ &\cup \{(\hat{x}, x) \mid x \in V_R\} \cup \{(\hat{x}, y) \mid i < j, x \in \tilde{f}(i), y \in \tilde{f}(j)\} \end{aligned}$$

We give a representation of a partial game in Figure 7.3. Once again, there is a clear bijection between the strategies of G and of $G(\tilde{f})$. For any MAX strategy σ, under best response, we thus have $v_\sigma(\hat{x}) \leq v_\sigma(\hat{y})$ for $x <_{\tilde{f}} y$. Hence, the partial order game forces an order on the random vertices. If f is a total order, then we say that $G(f)$ is a total order game.

Notice that if the order of the random vertices of game G are f under optimal strategies (σ^*, τ^*), then (σ^*, τ') is also optimal in $G(f)$, where $\tau'(x) = \tau^*(x)$ for vertices x not in V_R^{MIN} and $\tau'(\hat{x}) = x$ for $\hat{x} \in V_R^{\text{MIN}}$. Indeed, the random stopping condition implies that each random vertex with positive value has an edge towards a vertex with higher value or the sink 1. Thus, adding V_R^{MIN} does not increase the null set, and we conclude using the optimality conditions of Lemma 1.28.

7.2.3 . Partial Order Games as Face of the Permutahedron

The faces of the permutahedron Π_r correspond to partial orderings of $\{1, \dots, r\}$. Indeed, let $\mathcal{F}(I, f)$ be a face of Π_r. Let $|I| = k$ and $I = \{i_1, \dots, i_k\}$ with $i_1 < \dots < i_k$. We write $i_0 = 0$ and $i_{k+1} = r$ and we consider the following partition:

$$\forall 1 \leq j \leq k+1,\ \tilde{f}(j) = \{f(t) \mid i_{j-1} < t \leq i_j\}.$$

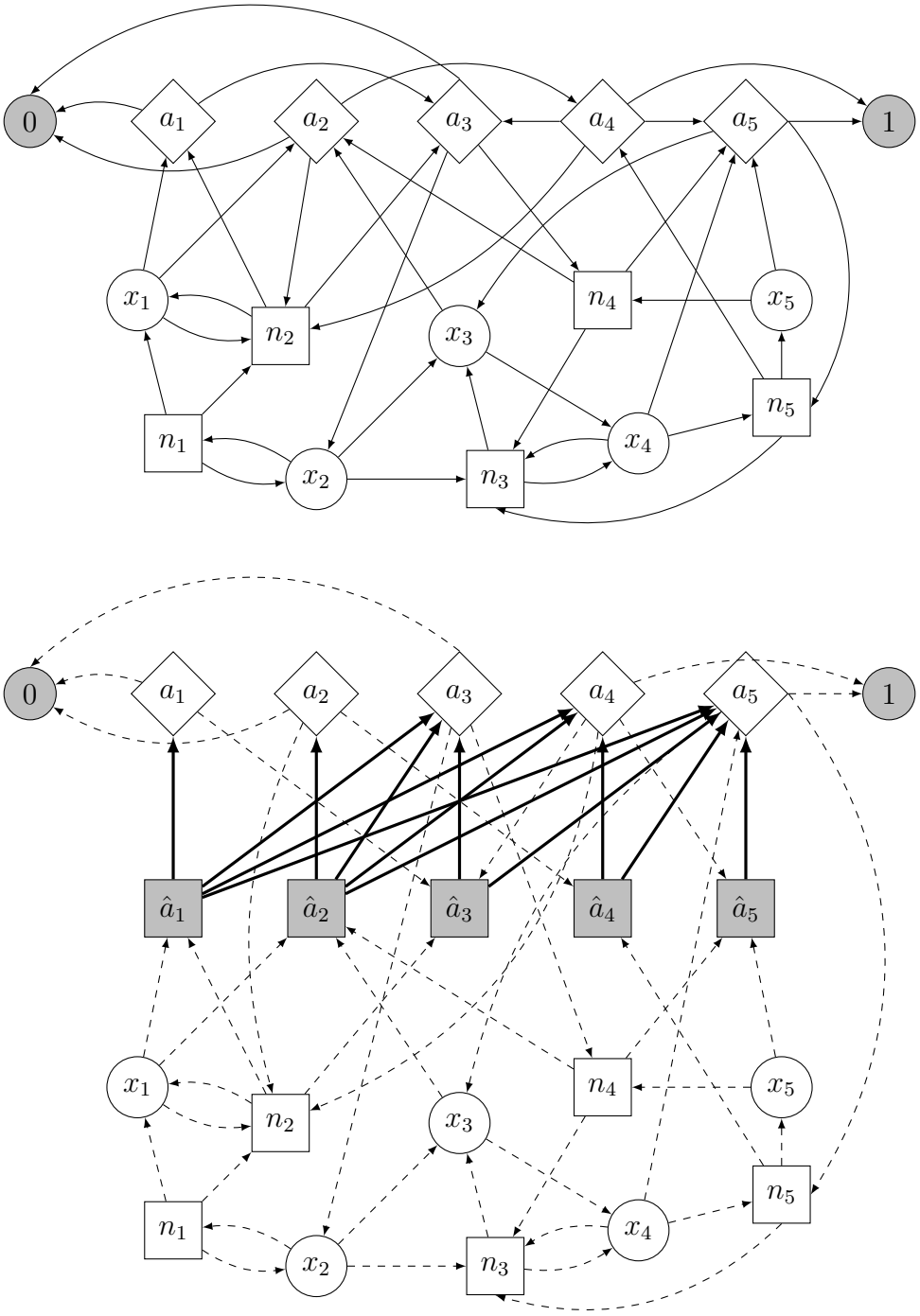


Figure 7.3: Partial order game with associated order $1, 2 < 3, 4 < 5$

Conversely, for $\tilde{f}$ a partial ordering with partition $\mathcal{P} = \left(\tilde{f}(i)\right)_{1\leq i\leq k}$, we consider the faces $\mathcal{F}(I, f)$ with f a total order coherent with $\tilde{f}$ and:

$$I = \{|\tilde{f}(1)|, |\tilde{f}(1)| + |\tilde{f}(2)|, \ldots, |\tilde{f}(1)| + \ldots + |\tilde{f}(k)|\}.$$

Hence, we have a bijection between the faces of the permutahedron and the partial order. Moreover, we notice that Π_r corresponds to the partial order $\tilde{f}(1) = S_r$ and each vertex Π_r is a face of dimension 0 corresponding to a total ordering. Thus, for $\tilde{f}$ a partial order, we write $\mathcal{F}(\tilde{f})$ the associated face and for $\mathcal{F}$ a face of Π_r we write $\tilde{f}(\mathcal{F})$ the associated partial order.

Let G be a stopping SSG with r random vertices. We consider an arbitrary order on the random vertices. For x and y in V_R, $x < y$ means that x is lesser than y in this arbitrary order. We consider the permutahedron Π_r and we associate to each face $\mathcal{F}$ the partial game $G\left(\tilde{f}(\mathcal{F})\right)$. We now want to define the orientation of the edges and prove that this orientation satisfies the unique sink orientation condition.

7.2.4 . The Value of Partial Order Game

We consider the optimal value associated to each partial order game. This will allow us to compare face and also nodes of the permutahedron.

Definition 7.10 (Value of Partial Order)**.** *Let G be an SSG and $\tilde{f}$ be a partial order, the value of the partial order $\tilde{f}$ written $v_{\tilde{f}}$ is defined as:*

$$v_{\tilde{f}} = v_{\sigma^*}^{G(\tilde{f})}$$

where σ^* is an optimal strategy of $G(\tilde{f})$.

Lemma 7.11. *Let f be a total ordering, the value of v_f can be computed in polynomial time.*

Proof. We recall that under optimal strategies, for x and y two vertices of V_R such that $f(x) < f(y)$, we have $v_f(\hat{x}) \leq v_f(\hat{y})$. Moreover, a path starting from a MAX vertex and reaching a random vertex has to first pass through a vertex of V_R^{MIN}.

Hence, we can consider σ a f-strategy corresponding to the order f on vertices of V_R^{MIN}. Let τ^* be an optimal MIN strategy of $G(f)$. We notice that v_f is the value vector of (σ, τ^*) using Lemma 1.22. Hence, σ is an optimal MAX strategy of $G(f)$ that can be computed in polynomial time with the attractor algorithm [GH08, AHMS08] and the computation of the best response. □

Similarly as in [GH08], a partial order defines an induced order.

Definition 7.12 (Induced Order)**.** *Let $\tilde{f}$ be a partial order, the induced order $\phi^{\tilde{f}}$ on V_R is defined as:*

$$\phi^{\tilde{f}}(x) < \phi^{\tilde{f}}(y) \iff v_{\tilde{f}}(x) < v_{\tilde{f}}(y) \text{ or } \left(v_{\tilde{f}}(x) = v_{\tilde{f}}(y) \text{ and } x < y\right)$$

Proposition 7.13. *Let $\tilde{f}$ be a partial ordering of V_R. If $\phi^{\tilde{f}}$ is coherent with $\tilde{f}$ then $v_{\tilde{f}} = v^G$.*

Proof. We suppose that $\tilde{f}$ is a partial ordering of G such that $\phi^{\tilde{f}}$ is coherent with $\tilde{f}$. It implies that for every $x \in V_R$ we have $v_{\tilde{f}}(x) = v_{\tilde{f}}(\hat{x})$. Let (σ^*, τ^*) be a pair of optimal strategies of $G(\tilde{f})$. Since G is random stopping, v_f is the value vector of (σ^*, τ^*) in G and they are optimal strategy by Lemma 1.28. □

7.2.5 . Adjacent Total Order Game

In this section, we define an orientation of the vertices of the permutahedron such that it satisfies the unique sink condition. Let f and f' be two adjacent vertices of Π_r. This means that there exists i such that $f' = f \circ (i\ i{+}))$. In order to ease the notation, similarly as for the cubes, we write $f' = \overline{f}^i$. We want to define an orientation from f to f' if $v_f < v_{f'}$. We show that it can be done by only evaluating v_f. We recall that we denote by $f(i)$ the i-th random vertices and $\hat{f}(i)$ the corresponding MIN vertices in V_R^{MIN} under total order f.

Lemma 7.14 (Induced Order Switch)**.** *Let f be a total ordering and let i be such that $\phi^f(f(i)) > \phi^f(f(i+1))$, then $v_f \leq v_{\overline{f}^i}$.*

Proof. We start by considering the game G' which is a copy $G(f)$ with the added edge $\left(\hat{f}(i+1), f(i)\right)$. We recall that $v_f(f(i)) \geq v_f(\hat{f}(i)) = v_f(\hat{f}(i{+}1)) \leq v_f(\hat{f}(i{+}2))$. Moreover, since G is random stopping, it implies that every random vertex with positive value has an edge towards a vertex of higher value. Thus, no attractor can be constructed with the added edges and for (σ^*, τ^*) a pair of optimal strategies of $G(f)$, they are also optimal in G'. We now remove from G' the edge $\left(\hat{f}(i), f(i+1)\right)$. This construct the game $G\left(\overline{f}^i\right)$ and we notice that every strategy τ of $G\left(\overline{f}^i\right)$ is also a strategy of G'. Writing, $\Sigma^{\text{MIN}}(G)$ the set of positional MIN strategies of G, it results:

$$\forall \sigma, \inf_{\tau \in \Sigma^{\text{MIN}}(G')} v^{G'}_{\sigma,\tau} \leq \inf_{\tau \in \Sigma^{\text{MIN}}(G(\overline{f}^i)} v^{G'}_{\sigma,\tau}$$

Hence, it results that $v_f \leq v_{\overline{f}^i}$ □

We thus define the following orientation $\mathcal{O}$. For f a vertex of Π_r, $\mathcal{O}(f, \overline{f}^i) = \overline{f}^i$ if and only if $\phi^f(f(i)) > \phi^f(f(i+1))$. This defines an orientation, since ϕ^f defines a total order.

Theorem 7.15. *The orientation $\mathcal{O}$ satisfies the unique sink orientation property.*

Proof. Let $\mathcal{F} = (I, f_0)$ be a face of Π_r with associated partial order $\tilde{f}$. Let σ be an optimal MAX strategy of $G(\tilde{f})$ and let f^* be the order of the vertices of V_R^{MIN} considering the extension of $\tilde{f}$ to a total order using the above defined arbitrary order on random vertices. Hence, f^* is an extension of $\tilde{f}$. By definition of our order, it implies that f^* is a sink of $\mathcal{F}$. Moreover, any order f coherent with $\tilde{f}$ that satisfies $v_f = v_{\tilde{f}}$ has a path towards f^* since either they are already ordered following the order of random vertices or not, in which case they can be sorted using bubble sort.

Now, let us consider a sink f of $\mathcal{F}$. It implies that for every $i \notin I$, $\phi^f(f(i)) < \phi^f(f(i+1))$. Let σ be an optimal strategy of $G(f)$. There is a best response τ that does not use any edges $(\hat{x}, y)$ with x and y in $\tilde{f}(k)$ for some k. Thus, they can be removed, to obtain $G\left(\tilde{f}\right)$ and (σ, τ) is a pair of optimal strategy of $G\left(\tilde{f}\right)$. Hence, $v_f = v_{\tilde{f}}$ and it results that $\mathcal{F}$ has a unique sink. □

7.3 . The USO Problem on the Permutahedron

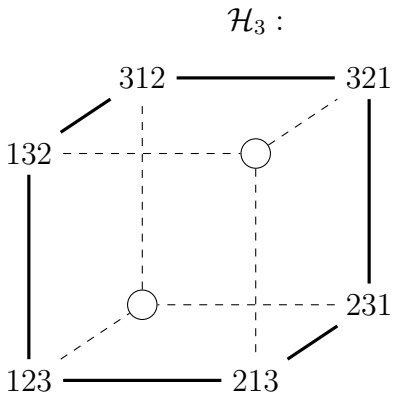


Figure 7.4: The permutahedron Π_3 embedded in the cube $\mathcal{H}_3$. We notice that $\mathcal{H}_3 \setminus \Pi_3$ is not connected.

In all this section, we consider the specificity of solving a USO problem on the permutahedron when compared to solving it on grids. For the time being, the fastest known algorithm on the permutahedron consists in evaluating each even permutation which gives the orientation of all edges of the permutahedron. Moreover, this is optimal in the case of finding a sink in Π_3. Indeed, in Π_3, there is exactly one source and one sink. We can consider the following adversary that places the source in the first evaluation and always orients the edges towards where the most amount of consecutive vertices have not been evaluated.

7.3.1 . The Permutahedron as a Partial Cube

An idea to solve the USO problem on the permutahedron is to embed the permutahedron in a cube. This is possible since the permutahedron is **a partial cube**. (See [Ovc11] for more details on partial cubes)

Definition 7.16 (Partial Cube)**.** *A Partial Cube is the induced subgraph of an hypercube.*

Lemma 7.17. *The permutahedron Π_r is a partial cube.*

Proof. One can consider the hypercube $\mathcal{H}$ of dimension $r(r-1)/2$. A coordinate of $\mathcal{H}$ is a pair of vertices of V_R and the associated edges an ordering of them. The permutahedron is the subgraph induced by the vertices of $\mathcal{H}$ that verify a total ordering of V_R. □

We represent how Π_3 is embedded in $\mathcal{H}_3$ in Figure 7.4. It is important to notice that the graph $\mathcal{H}$ without the vertices of Π_r is not connected, hence the USO problem on Π_r cannot easily be translated into a USO problem on $\mathcal{H}$ by orienting each edges from $\mathcal{H} \setminus \Pi_r$ towards Π_r. Indeed, such a transformation would create one source by connected component, which contradicts the uniqueness of the source of Lemma 6.16.

Moreover, using the Fibonacci Seesaw Algorithm to solve a USO problem on $\mathcal{H}$ cost $O\left(1.61^{r(r-1)/2}\right)$ while visiting each vertex of the permutahedron costs $O(r!) = o(r^r) = o\left(1.61^{r(r-1)/2}\right)$.

7.3.2 . Properties on Cubes are not True on Permutahedrons

Some properties of USO on cubes or grids are not true on the permutahedron. The first one being the uniqueness of the switch set of each vertex. First, one can notice that each vertex has exactly $r-1$

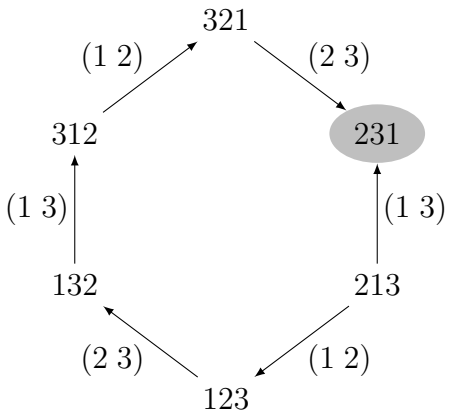


Figure 7.5: In this instance, both 123 and 321 have the same switch set.

neighbours, hence there are at most 2^{r-1} different switch sets which is less than the $r!$ vertices of Π_r. Another possibility is to consider for switch set the different transposition. In other words, this corresponds to the edges of the extended cube presented in Section 7.3.1. Once again, this does not ensure uniqueness, as we can see in Figure 7.5.

Moreover, taking the quotient of the permutahedron by a face does not give another permutahedron. Indeed, the face of the permutahedron are not permutahedron.

Finally, while randomised strategies for binary SSG can be viewed as interior points of the hypercube, such interpretation of interior points does not hold for the permutahedron.

Although modelling SSGs as a USO problem on the permutahedron does not give new algorithms, we hope that it can help in studying algorithm on f-strategies and be used to refine the upper bound on the Gimbert and Horn's Algorithm.

8 - Impartial SSG and Slowed Shortest Time

Contents

In this chapter, we study SSG defined implicitly and more specifically, SSG defined as a sum of SSG. In combinatorial game theory, the sum of two games is the game where players can play in either one of the two games. We define Impartial Simple Stochastic Games (ISSG) that construct SSG that can easily be summed. We show that the value of the sum of ISSG can not be easily expressed using the value of the added games, even in a more restricted version of ISSG that we call skipping games.

8.1 . Impartial Game

8.1.1 . Definition and Sum of Games

In combinatorial game theory, an impartial game is a perfect information turned-based two-player game where the possible moves depend only on the current position and not on whose player turn it is. In all this chapter, we will only consider acyclic impartial game, even if it is not specified. We start by giving a brief introduction on impartial games and combinatorial game theory. For more in-depth information, one can see [BCG04, Con00].

Definition 8.1 (Impartial Game)**.** *An impartial game is an acyclic directed graph $G = (V, E)$.*

The game is played as follows. A token is placed on some vertices of the graph and is moved alternatively by both players alongside the edges. The player that cannot move the token lose. Impartial game are trivially acyclic SSG and can be solved by graph traversal. Thus, they are often described by a definition smaller than the game size. One of the most classical instance of Impartial Games are subtraction games where several piles of tokens are placed between two players that alternatively removes some tokens according to some rules (how many they want in a single piles, at most k token in totals, the same number of tokens across several piles...). An instance of impartial game, called the Wythoff's game, is presented in Figure 8.1.

More often than not, combinatorial games are defined recursively as a list of move to other games. Thus, the starting position is given in the definition of the game. In the rest of the chapter, if the context is clear enough, we will make no distinction between talking about a position or vertex x of a game G and the game G to describe its initial position. Thus, for an impartial game G, we can write $G = \{G_1, \dots, G_k\}$ where $G_1, \dots, G_k$ are the reachable games from G. The game that starts in a vertex with out-degree 0 is said to be terminal and is noted $\mathbf{0}$ or $\{\}$ since there is no move from it. Hence, using graph formalism, G is a vertex defined as the set of its out-neighbourhood.

Definition 8.2. *For G an impartial game, we create a partition of the vertices of G in $\mathcal{P}$-position and $\mathcal{N}$-position as follows. Every terminal vertex, the vertices with out-degree 0, is a $\mathcal{P}$-position. If a vertex has at least one $\mathcal{P}$-position in its out-neighbourhood, it is a $\mathcal{N}$-position. If a non-terminal vertex does not have any $\mathcal{P}$-position in its out-neighbourhood, it is a $\mathcal{P}$-position.*

The $\mathcal{N}$-position are also called non-$\mathcal{P}$-position. As shown in the next lemma, this partition characterise the winning position for both player

Lemma 8.3. *If a game starts in a $\mathcal{P}$-position, the second player has a strategy to win. If a game starts in a $\mathcal{N}$-position, the first player has a strategy to win the game.*

Proof. This can be proven by induction. Starting a game in a terminal state is an automatic loss for the first player, and terminating vertices are $\mathcal{P}$-position. If the game starts in a $\mathcal{N}$-position, the first

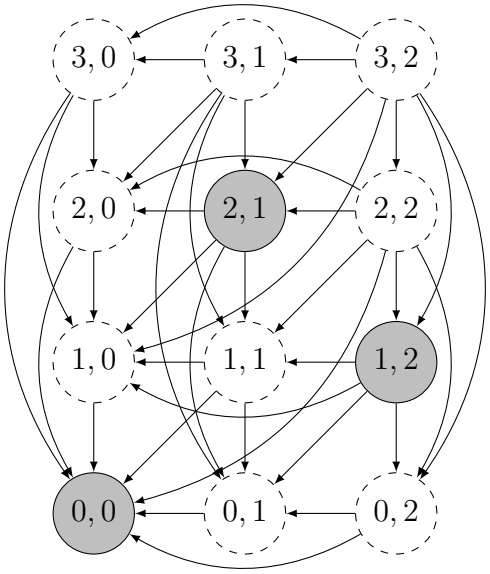


Figure 8.1: Representation of the Wythoff's game starting in position $(3, 2)$. The gray vertices represent the $\mathcal{P}$-positions and the dashed vertices the $\mathcal{N}$-positions.

player can move to a $\mathcal{P}$-position and, by induction, the second player loses the game. Finally, if the play start in a non-terminal $\mathcal{P}$-position, the first player has no other choice than to go to a $\mathcal{N}$-position from where the second player has a winning strategy. □

An instance of the partition in $\mathcal{P}$-positions and $\mathcal{N}$-positions is presented in Figure 8.1. The player which starts in a $\mathcal{P}$-position is called the losing player and the other one is called the winning player.

One of the main interests of studying combinatorial games, and more specifically in our case impartial games, is the possibility to sum games. Which allows us to study games by looking at smaller components.

Definition 8.4. *For $G = \{G_1, \ldots, G_k\}$ and $H = \{H_1, \ldots, H_l\}$ two impartial games, the sum of G and H, denoted by $G + H$, is the game*

$$G + H = \{G_1 + H, \ldots, G_k + H, G + H_1, \ldots, G + H_l\}$$

Notice that G and H can be empty.

In other words, the game $G + H$ is the game where players can choose to play either in G or in H. Thus, $G + H$ corresponds to the graph $G \square H$, the Cartesian product of G and H (see Definition 6.8). A representation of a sum is presented in Figure 8.2.

8.1.2 . The Sprague-Grundy Theorem

If G and H are both $\mathcal{P}$-positions, then $G+H$ is a $\mathcal{P}$-position. If G is a $\mathcal{P}$-position and H is a $\mathcal{N}$-position, then $G + H$ is a $\mathcal{N}$-position. The question is what happens if both G and H are $\mathcal{N}$-positions. In order to answer this question, one can use the Sprague-Grundy theorem (Theorem 8.8) proven independently by Sprague in [SPR35] and Grundy in [Gru39]. First, we need to recall the definition of two mathematical operators: the mex and the bitwise XOR. The definition of XOR is given in Definition 4.7.

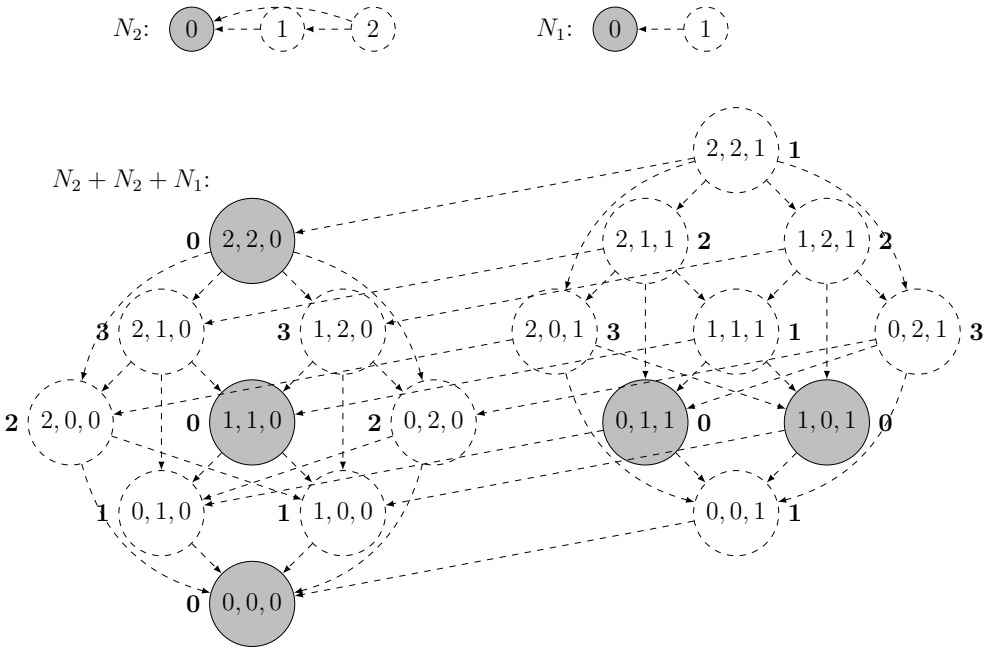


Figure 8.2: Representation of the Nim games N_2 and N_1 and representation of $N_2 + N_2 + N_1$. The grey vertices represent $\mathcal{P}$-positions. The bold number represents the Grundy value of the vertices.

Definition 8.5 (mex)**.** *For a finite set of non-negative integer E, the* mex *of E is the smallest natural number not in the set.*

For instance, the mex of $\{1, 3, 7\}$ is 0 and the mex of $\{0, 1, 2, 3\}$ is 4.

Definition 8.6. *The Grundy value or Nim value of a game $G = \{G_1, \ldots, G_k\}$ is defined as $g(\mathbf{0}) = 0$ and $g(G) = \operatorname{mex}\{g(G_i) \mid 1 \leq i \leq k\}$.*

The Grundy value of a game is presented in Figure 8.2.

Lemma 8.7. *A game G is a $\mathcal{P}$-position if and only if $g(G) = 0$.*

Proof. By definition of the mex, a game as Grundy value 0 if and only if none of its out-neighbours have Grundy value 0. This is the same construction as the $\mathcal{P}$-position. Hence, a game G is a $\mathcal{P}$-position if and only if $g(G) = 0$. □

Theorem 8.8 (Sprague-Grundy theorem [BCG04, Con00])**.** *For G and H two impartial games, $g(G + H) = g(G) \oplus g(H)$ where $\oplus$ is the bitwise operator XOR.*

Proof. We proceed by induction. If $G = \mathbf{0}$ and $H = \mathbf{0}$ then $G + H = \mathbf{0} + \mathbf{0} = \mathbf{0}$, $g(G) = 0$, $g(H) = 0$ and $g(G + H) = 0 = 0 \oplus 0$. Let $G = \{G_1, \ldots, G_k\}$ and $H = \{H_1, \ldots, H_l\}$. We want to show that $g(G) \oplus g(H) = \operatorname{mex} E$ where,

$$E = \{g(G_1) \oplus g(H), \ldots, g(G_k) \oplus g(H), g(G) \oplus g(H_1), \ldots, g(G) \oplus g(H_l)\}$$

First, we show that $g(G) \oplus g(H)$ is not in E. For all $i \leq k$, $g(G_i) \neq g(G)$ by the definition of mex and for all $j \leq l$, $g(H_i) \neq g(H)$. Thus, $g(G_i) \oplus g(H) \neq g(G) \oplus g(H)$ and $g(G) \oplus g(H_j) \neq g(G) \oplus g(H)$.

Let $x < g(G) \oplus g(H)$, we show that x is in E. We consider the position i of the first bit where x and $g(G) \oplus g(H)$ are different. Since x is less than $g(G) \oplus g(H)$, it implies that the i^{th} bit equal 0 for x and 1 for $g(G) \oplus g(H)$. By symmetry, we assume that this bit is equal to 1 in $g(G)$ and 0 in $g(H)$. Notice, that the $i-1^{st}$ bits of $g(G) \oplus g(H)$ and x are equal, then if we XOR $g(H)$ on both sides they are still equals in the first $i-1$ bits. Hence, $g(H) \oplus x$ differs from $g(G)$ starting from the i^{th} bit and $g(H) \oplus x < g(G)$. By definition of the mex, there exists G' accessible from G such that $g(G') = g(H) \oplus x$. Hence, by induction hypothesis, $g(G' + H) = g(G') \oplus g(H) = x$ with $G' + H$ accessible from $G + H$. Thus, we have proven that

$$g(G + H) = g(G) \oplus g(H)$$

□

8.2 . Impartial Simple Stochastic Game

8.2.1 . Definition of ISSG

In order to sum SSGs, we introduce ISSG that are a randomise version of impartial games.

Definition 8.9 (Impartial Simple Stochastic Game)**.** *An Impartial Simple Stochastic Game (ISSG) is an acyclic directed graph G, together with:*

1. *A partition of the vertex set V in two parts V_P and V_R such that each vertex of V_R has at least one outgoing edge.*
2. *For every $x \in V_R$, a probability distribution $p_x(\cdot)$ with rational values, on the out-neighbourhood of x.*
3. *We authorise probabilistic self loop. In other words, if $x \in V_P$ is the only in-neighbour of $a \in V_R$, we authorise the edge (a, x) in E if $p_a(x) < 1$.*

The vertex of V_P are called playable vertex and the vertex of V_R are called random vertices. The game is played as follows. The two players take turns moving a token that start on some initial vertex x_0. If the token is in a vertex of V_R, it is randomly moved according to the probability distribution p_x. If the token is in a playable vertex with a positive out-degree, the player whose turn it is, moves the token alongside the edge, and it is now the turn of the other player. If the token is in a playable vertex with no outgoing edges, the player whose turn it is, has to pay 1 to the other player. This definition allows to easily sum ISSG and thus to construct large game with concise representation.

Remark 8.10. We notice that in our definition of sum of ISSG, the play continues until every added game ends. Another interesting concept to study is the case when the game stop when a sink is reached in any games. Such sum as for instance be studied for summing Markov chain [DTW03].

We keep the same definition of strategies and value vector in ISSG as in SSG. ISSG can be modelled using SSG. Hence, all properties on SSG remains on ISSG.

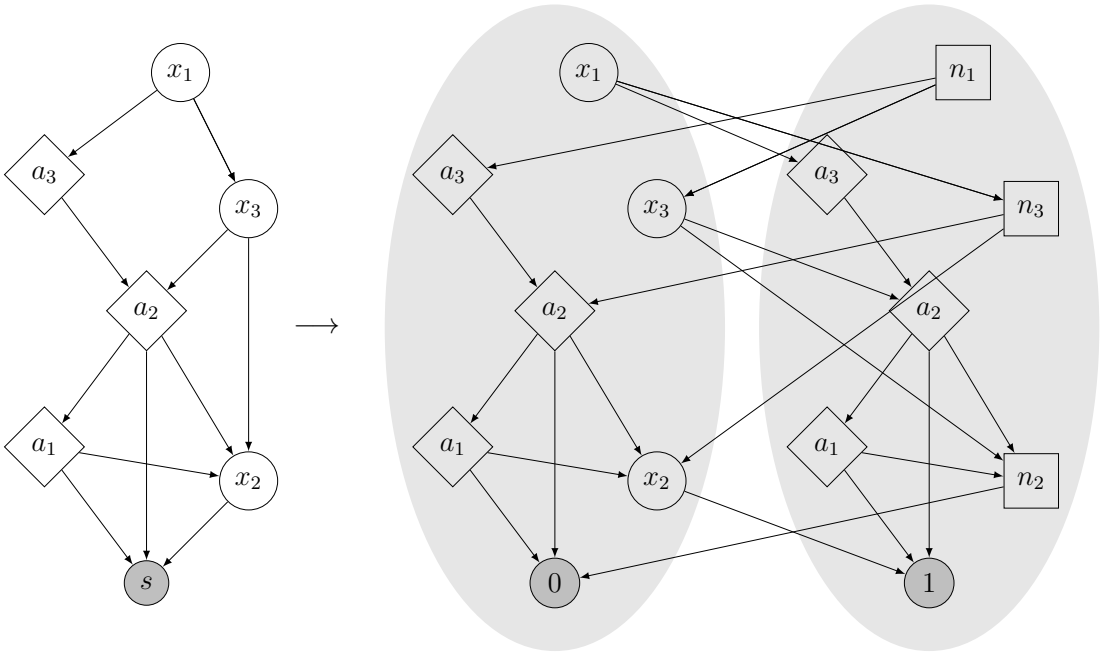


Figure 8.3: Reduction from ISSG to SSG

Proposition 8.11. *The problem of finding the value vector of an ISSG can be polynomially reduced to solving an SSG.*

Proof. Let $G = (V, E)$ be an ISSG. We construct, $G' = (V', E')$ an SSG that emulate G. Let $V' = V \times \{\text{MAX}; \text{MIN}\}$. Let $V_\circ$ be the set of vertices of V_P with no outgoing degree. $V'_S = V_\circ \times \{\text{MAX}; \text{MIN}\}$ with value 0 for the vertex of $V_\circ \times \{\text{MAX}\}$ and 1 for the other. $V'_R = V_R \times \{\text{MAX}; \text{MIN}\}$ and finally, V_{MAX} and V_{MIN} are respectively $V_P \setminus V_\circ \times \{\text{MAX}; \text{MIN}\}$ and $V_P \setminus V_\circ \times \{\text{MAX}; \text{MIN}\}$.

For every edge $e = (x, y)$ in E we have in E' the edge $(x \times \{\text{MAX}\}, y \times \{\text{MAX}\})$ and $(x \times \{\text{MIN}\}, y \times \{\text{MIN}\})$ if $x \in V_R$ and $(x \times \{\text{MAX}\}, y \times \{\text{MIN}\})$ and $(x \times \{\text{MIN}\}, y \times \{\text{MAX}\})$ otherwise.

The optimal vector value v of G satisfies

$$\forall x \in V,\ v(x) = 2v'(x \times \{\text{MAX}\}) - 1$$

with v' the optimal value vector of G'. □

The reduction is presented Figure 8.3

Theorem 8.12. *The optimal strategy of an ISSG does not depend on whose turn it is.*

Proof. Directly from the impartiality of the game. □

8.2.2 . One Stack Mistake Game

A natural consideration when looking at impartial game is to consider the case where one player makes a mistake, or when a bug occurs in the transmission of the action. At each turn, we consider a small probability of a mistake being made. We consider the following transformation of Impartial Game into ISSG.

During each move, the players have a probability p to play randomly along all possible edges. Let us define the **one stack mistake game**. There is a pile of tokens and each player takes as much token as they want, but at least one. Each time, there is a probability $1 > p > 0$ that a mistake is made and that the player removes a number of tokens uniformly between all possible choice. If a player cannot move, they lose the game.

The victory probability P_k is the probability of winning the game when starting with k tokens and both sides playing optimally. Hence, $P_0 = 0$. This game is clearly an acyclic impartial SSG where the set of vertices noted $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots\}$ correspond to the number of tokens left in the stack. Thus, the optimal strategies are positional, and the victory probability is defined by:

$$P_i = \max_{k \leq i-1} (1 - P_k)(1 - p) + \frac{p}{i} \sum_{j=0}^{j=i-1} (1 - P_j)$$

And the optimal strategy σ^* is defined by the following equality for all $i \geq 1$.

$$\sigma^*(\mathbf{i}) = \arg\max_{k \leq i-1} (1 - P_k)(1 - p) + \frac{p}{i} \sum_{j=0}^{j=i-1} (1 - P_j)$$

Since for all i, $P_i \geq 0$, the optimal strategy is for all $i \geq 1$:

$$\sigma^*(\mathbf{i}) = \mathbf{0}$$

Proposition 8.13. *The victory probability satisfies for all $i \geq 0$:*

$$P_i = \frac{1 - \frac{\Gamma(i-p)}{\Gamma(i+1)\Gamma(-p)}}{p+1}$$

where $\Gamma(\cdot)$ is the gamma function.

Proof. We define $F_i = \dfrac{1 - \frac{\Gamma(i-p)}{\Gamma(i+1)\Gamma(-p)}}{p+1}$. We show by induction on i that $F_i = P_i$.

$$F_0 = \frac{1 - \frac{\Gamma(-p)}{1 \cdot \Gamma(-p)}}{p+1} = 0 = P_0$$

We suppose that $P_i = F_i$ for some $i \geq 0$, and we now prove that $P_{i+1} = F_{i+1}$.

First, we present a recursive relation between P_{i+1} and P_i.

$$(i+1)P_{i+1} - iP_i = (1-p)(i+1) + p\sum_{j=0}^{j=i}(1 - P_j) - (1-p)i - p\sum_{j=0}^{j=i-1}(1 - P_j)$$

$$(i+1)P_{i+1} - iP_i = 1 - p + p(1 - P_i)$$

$$(i+1)P_{i+1} = 1 + (i-p)P_i$$

$$P_{i+1} = \frac{1}{i+1} + \frac{i-p}{i+1}P_i$$

By induction hypothesis, we obtain the following formula:

$$P_{i+1} = \frac{1}{i+1} + \frac{i-p}{i+1}\frac{1 - \frac{\Gamma(i-p)}{\Gamma(i+1)\Gamma(-p)}}{p+1}$$

Which is equal to

$$P_{i+1} = \frac{1}{i+1} + \frac{i-p}{(i+1)(p+1)} - \frac{\frac{(i-p)\Gamma(i-p)}{(i+1)\Gamma(i+1)\Gamma(-p)}}{p+1}$$

We know that for any $x > 0$, $x\Gamma(x) = \Gamma(x+1)$. Hence:

$$P_{i+1} = \frac{p+1+i-p}{(i+1)(p+1)} - \frac{\frac{\Gamma(i+1-p)}{\Gamma(i+2)\Gamma(-p)}}{p+1} = \frac{1 - \frac{\Gamma(i+1-p)}{\Gamma(i+2)\Gamma(-p)}}{p+1}$$

Thus, we have proven that $P_{i+1} = F_{i+1}$. □

8.2.3 . Skipping Game

We now consider another way to introduce randomness. It is natural to assume that sometimes the turn of one of the two players is skipped by mistake. Either they simply forget to play or they were not able to answer in a given timeframe. We consider an impartial combinatorial game, and we add the following rule: whenever a player has to make a move, there is a probability p that they forgot to play and that their turn is skipped. In other words, each action has probability p to stay in the same position.

Definition 8.14 (Skipping Game)**.** *The skipping version of an acyclic impartial game $G = (V, E)$, or skipping game, is the impartial SSG $G^S(V^S, E^S)$ where $V^S = V \bigcup E$ with V the playable vertices and E the random vertices. For each edge $e = (x, y) \in E$, there is (x, e), (e, y) and (e, x) in E^S. The associated probability distribution is $1 - p$ for (e, y) and p for (e, x).*

Definition 8.15. *For x a vertex of an acyclic impartial game, we call the slowed shortest time $\kappa(x)$ the number of steps of a strategy when the winning player wants to minimise the number of steps and the losing player wants to maximise it.*

We write $\kappa(G)$ the value of $\kappa(x_0)$ in game G where x_0 is the starting position of G.

Remark 8.16. The $\mathcal{P}$-position of the game have even slowed shortest time, while the other vertices have odd slowed shortest time. This is due to the fact that every optimal play alternates between $\mathcal{P}$-positions and $\mathcal{N}$-positions and that $\kappa(\mathbf{0}) = 0$.

We show that κ is useful to study ISSG.

Theorem 8.17. *For G^S the skipping version of game G, and x a playable vertex, the value vector of G^S is defined by:*

$$v(x) = -\left(-\frac{1-p}{1+p}\right)^{\kappa(x)}$$

Proof. We prove this result by induction on the depth of the acyclic graph of the game. First, the vertices with out-degree 0 have a slowed shortest time of 0 and value -1.

For x a $\mathcal{P}$-position of G with positive out-degree, the vertices of $N^+(x)$ the out-neighbourhood of x are $\mathcal{N}$-positions. By induction hypothesis, we have

$$\forall y \in N^+(x),\ v(y) = -\left(-\frac{1-p}{1+p}\right)^{\kappa(y)}$$

Since the optimal strategy is a positional strategy that satisfies the optimality condition of the value vector (see Lemma 1.28, the value of vertex x can be described.

$$v(x) = \max_{y \in N^+(x)} -(1-p)\left(-\left(-\frac{1-p}{1+p}\right)^{\kappa(y)}\right) - pv(x)$$

We recall that $\kappa(y)$ have odd value for all $y \in N^+(x)$. Moreover, $0 < \dfrac{1-p}{1+p} < 1$ since $0 < p < 1$. The value $v(x)$ is maximised by choosing y that minimise $v(y)$ thus by choosing y with the highest $\kappa(y)$. In addition, since x is a $\mathcal{P}$-position and the losing player wants to maximise the length of the play, we have $\kappa(x) = 1 + \max_{y \in N^+(x)} \kappa(y)$. Thus:

$$v(x) = (1-p)\left(-\frac{1-p}{1+p}\right)^{\kappa(x)-1} - pv(x)$$

$$(1+p)v(x) = (1-p)\left(-\frac{1-p}{1+p}\right)^{\kappa(x)-1}$$

$$v(x) = -\left(-\frac{1-p}{1+p}\right)^{\kappa(x)}$$

This proves the induction property for $\mathcal{P}$-position.

For x a $\mathcal{N}$-position, by definition, there is at least one $\mathcal{P}$-position in $N^+(x)$. By induction hypothesis, its $\mathcal{P}$-position neighbours have negative value and its non-$\mathcal{P}$-position neighbours have positive value. The optimal strategy in position x is to go to the vertex with minimal value. Thus, the optimal strategy is to choose the $\mathcal{P}$-position y with smallest slowed shortest time. In addition, $\kappa(x) = \kappa(y) + 1$ by definition. Hence:

$$(1+p)v(x) = (1-p)\left(-\frac{1-p}{1+p}\right)^{\kappa(x)-1}$$

and

$$v(x) = -\left(-\frac{1-p}{1+p}\right)^{\kappa(x)}$$

This proves the induction property for non-$\mathcal{P}$-position. □

Thus, we can compute the value of the skipping version of impartial games only by knowing their slowed shortest time. Another reason to study κ is the case of amortized gain in impartial games. Let $0 < q < 1$. We consider an impartial game where the losing player has to pay q^t to the winning player, where t is the length of the game. In this case, the gain starting in the $\mathcal{N}$-position x is q^x.

8.3 . Computing the Slowed Shortest Time for Some Classic Games

In this section, we compute κ for some well-known impartial game.

8.3.1 . Nim Games

The first game that we study is the classical Nim game.

Definition 8.18. *The Nim game N_k is the graph $G = (V, E)$ with $V = \{0, \ldots, k\}$ and with $E = \{(i, j) \mid i > j\}$.*

In other words, the game consists of k tokens, with each player removing in turn as many tokens as they want (at least one). The player that cannot remove a token lose. Obviously, in this game, the optimal strategy consists in taking all tokens. The Grundy value of N_k is k. Originally, the Sprague-Grundy theorem defines the Grundy value of a game G as the size of a one heap Nim game equivalent to G. This is why the Grundy value is also called Nim value.

Lemma 8.19. *For a Nim game N_k and $1 < k$, $\kappa(N_k) = 1$.*

Proof. There is only one $\mathcal{P}$-position thus the slowed shortest time of $\mathcal{N}$-positions in N_k is 1. □

We denote by $N_{k_1+\ldots+k_n}$ the game $N_{k_1} + \ldots + N_{k_n}$. A representation of N_{2+2+1} is given in Figure 8.2. The goal is to compute the κ function of $N_{k_1+\ldots+k_n}$.

Remark 8.20. By Theorem 8.8, the $\mathcal{P}$-positions of $N_{k_1+\ldots+k_n}$ are the vertices $(x_1, x_2, \ldots, x_n)$ such that $x_1 \oplus x_2 \oplus \ldots \oplus x_n = 0$ where $\oplus$ is the bitwise XOR operator in binary. We denote by K_n the set of such vertices and K_n^* the set $K_n \setminus \{(0, 0, \ldots, 0)\}$.

Proposition 8.21. *For $n \geq 2$, $(k_1, k_2, \ldots, k_n) \in K_n^*$ we have*

$$\kappa\left(N_{k_1+\ldots+k_n}\right) = \sum_{i=1}^{n} k_i$$

Proof. Let start in position $(k_1, k_2, \ldots, k_n) \in K_n^*$. We consider the index i such that k_i have the lowest $2-$adic valuation. We recall that the 2-adic value of x is the largest i such that 2^i divides x. In other words, we consider the stack whose number of tokens has the rightmost one in binary form, in position j. Since the binary XOR of every stack is null, we have at least another stack that has the same $2-$adic valuation. The first player removes one token in k_i. The winning strategy for the second player is to reach another position in K_n. Hence, they must remove tokens in a heap such that in binary form the j rightmost digits are changed and not the others. By minimality of the 2-adic valuation of k_i, the second player has to remove one token from one of the heaps that had same 2-adic valuation as k_i.

In every $\mathcal{P}$-position, the losing player has a play removing one token that forces the second player to also remove only one token. Moreover, the maximal number of steps of any play in $N_{k_1+\ldots+k_n}$ is $\sum_{i=1}^{n} k_i$, hence we have:

$$\kappa\left(N_{k_1+\ldots+k_n}\right) = \sum_{i=1}^{n} k_i$$

□

Notice that there exists $\mathcal{P}$-positions where removing only one token allows the winning player to remove more than one token. For instance, in N_{6+4+2}, we have $6 = 110_2$, $4 = 100_2$ and $2 = 010_2$. If the first player chose to remove one in the second heap to go in N_{6+3+2}, then the other player can remove 5 tokens in the first stack to reach N_{1+3+2}. $1 \oplus 3 \oplus 2 = 0$ and is a $\mathcal{P}$-position.

8.3.2 . Bounded Nim Games

A classic variant of Nim games are bounded Nim games. In those games, each player can take a limited number of tokens.

Definition 8.22. *For m a positive integer, the game $N_k^{(m)}$ is the graph $G = (V, E)$ with $V = \{0, \ldots, k\}$ and with $E = \{(i, j) \mid i > j \geq \max(i - m, 0)\}$.*

Lemma 8.23. *The Grundy value of $N_k^{(m)}$ is $k\%(m+1)$ where $k\%(m+1)$ is the remainder in the Euclidean division of k by $m+1$.*

Proof. By immediate induction using the mex definition. □

Computing κ for a bounded Nim game is easy. We just have to notice that using the Grundy value computed Lemma 8.23, from every $\mathcal{N}$-position there is only one accessible $\mathcal{P}$-position and thus the $\mathcal{P}$-position cannot be skipped. We have the following formula:

$$\forall k \in \mathbb{N},\ \kappa\left(N_k^{(m)}\right) = 2\left\lfloor \frac{k}{m+1} \right\rfloor + \mathbb{1}_{(m+1)\nmid k}$$

where $\mathbb{1}_{(m+1)\nmid k}$ equals 1 if $m+1$ does not divide k and 0 otherwise. Here $\left\lfloor \frac{k}{m+1} \right\rfloor$ counts the number of non-terminal $\mathcal{P}$-position that will be visited during the game and $\mathbb{1}_{(m+1)\nmid k}$ counts if the starting position is a $\mathcal{P}$-position or not.

Theorem 8.24. *Let $n \geq 2$ and $m_1, \ldots, m_n$ and $k_1, \ldots, k_n$ be integers greater than 1. Let K be the set of $\mathcal{P}$-positions of $N_{k_1}^{(m_1)} + \ldots + N_{k_n}^{(m_n)}$.*

$$\forall x = (x_1, \ldots, x_n) \in K,\ \kappa(x) = \sum_{i=1}^{n} x_i\%(m_i+1) + 2\sum_{i=1}^{n} \left\lfloor \frac{x_i}{m+1} \right\rfloor$$

Proof. We proceed by induction on the depth of the game to prove that for each $\mathcal{P}$-position x, $\kappa(x)$ is equal to $\phi(x)$ where $\phi(x)$ is our above formula. If there is no token in any pile, then we have $\kappa((0, \ldots, 0)) = 0$.

Let x be a non-empty $\mathcal{P}$-position. We consider i such that $g(x_i) = x_i\%(m_i+1)$ has the smallest 2-adic value. We recall that by convention, the 2-adic value of 0 is $+\infty$. First, we assume that $g(x_i) \neq 0$. We consider the case where the losing player takes a token from the pile i. The winning player can choose to play in the same pile. In this case, they must take m_i tokens in order to return to a $\mathcal{P}$-position x' and the number of moves is by induction $\phi(x') + 2 = \phi(x)$. They can also play elsewhere. If they choose a stack that has the same 2-adic number than x_i, they have to take a single token in order to reach a $\mathcal{P}$-position x'. By induction, we have $\phi(x') + 2 = \phi(x)$. Lastly, they can choose to play somewhere with a higher 2-adic number. In this configuration, in order to reach a $\mathcal{P}$-position x',

the Grundy number associated with this stack needs to increase strictly. Indeed, the leftmost digits have to stay the same. However, we decrease by one, one of the quotients. Thus $\phi(x') \geq \phi(x) - 2 + 1$. Since the winning player wants to minimise $\kappa(x)$, they will choose one of the first two options and the play will last $\phi(x)$ moves. If $g(x_i) = 0$ similarly, the winning player has to play in x_i to minimise the length of the game. Thus $\kappa(x) \geq \phi(x)$.

We now prove that $\kappa(x) \leq \phi(x)$. For each stack i where the losing player takes k tokens where there are more than m_i tokens, the next player can always choose to take $m_i + 1 - k$ tokens and assure by induction that the game last $\phi(x') + 2 = \phi(x)$ turns. Otherwise, if the losing player decreases the number of tokens in a pile with $x_i \leq m_i$ tokens, there is a stack whose Grundy value can be decreased to reach a $\mathcal{P}$-position x'. Thus $\phi(x') \leq \phi(x) - 2$. Hence, $\kappa(x) \leq \phi(x)$.

Thus, we have proven that $\kappa(x) = \phi(x)$. □

Corollary 8.25. *In the game* $N_{k_1}^{(m_1)} + \ldots + N_{k_n}^{(m_n)} + N_{k'_1} + \ldots + N_{k'_{n'}}$ *with* $\mathcal{P}$*-position* K*, for* $x = (x_1, \ldots, x_n, x'_1, \ldots, x'_{n'}) \in K$ *we have*

$$\kappa(x) = \sum_{i=1}^{n} x_i \% (m_i + 1) + 2 \sum_{i=1}^{n} \left\lfloor \frac{x_i}{m+1} \right\rfloor + \sum_{i=1}^{n'} x_i$$

Proof. A Nim game of k token is equivalent to a bounded Nim game $N_k^{(k)}$. Thus, this follows from Theorem 8.24. □

We can reformulate this result using the Grundy value and κ value of Nim and bounded Nim games.

Corollary 8.26. *Let* $G_1, \ldots, G_n$ *be* $n \geq 2$ *either Nim or bounded Nim games. If* $G_1 + \ldots + G_n$ *is a* $\mathcal{P}$*-position then:*

$$\kappa(G_1 + \ldots + G_n) = \sum_{i=1}^{n} g(G_i) + \kappa(G_i) - \mathbb{1}_{g(G_i)>0}$$

where $\mathbb{1}_{g(G_i)>0} = 1$ *if* $g(G_i) > 0$ *and 0 otherwise.*

Proof. This is a direct consequence of Corollary 8.26 from the κ function and Grundy value of Nim and Nim bounded games. □

8.3.3 . Wythoff's Game

The Wythoff's Game is another subtracting game played on two stacks of tokens. A valid move is taking at least one token in one or the two piles. When a player removes from both stacks, they must take the same number of tokens in each stack.

Definition 8.27 (Wythoff's Game)**.** *The Wythoff's Game* $W_{i,j}$ *corresponds to the graph* $G = (V, E)$ *with* $V = \{0, \ldots, i\} \times \{0, \ldots, j\}$ *and*

$$\begin{aligned} E = &\{((i,j),(k,j)) \mid 0 \leq k < i\} \cup \{((i,j),(i,k)) \mid 0 \leq k < j\} \\ &\cup \{((i,j),(k,l)) \mid 0 \leq k < i, 0 \leq l < j, i - j = k - l\} \end{aligned}$$

A representation of Wythoff's game starting in position $(3, 2)$ is presented in Figure 8.1. The $\mathcal{P}$-positions of this game have been found by Wythoff in 1907 [Wyt07]. We consider the two sequences $(n_k)_{k\geq 0}$ and $(m_k)_{k\geq 0}$ defined by:

$$n_k = \lfloor k\varphi \rfloor$$
$$m_k = \lfloor k\varphi^2 \rfloor = n_k + k$$

where $\varphi = \dfrac{1+\sqrt{5}}{2}$ is the golden ration. The $\mathcal{P}$-positions are the (n_k, m_k) and (m_k, n_k). We call those the k^{th}-$\mathcal{P}$-position.

The original proof is done by saying that two $\mathcal{P}$-positions cannot have the same number of token in the same stack, and that the difference between the number of tokens in two $\mathcal{P}$-positions must be different. The $\mathcal{P}$-positions can thus be inductively constructed by saying that $(0, 0)$ is in K and that the next $\mathcal{P}$-position is the couple (a, b) and (b, a) where a is the smallest number not yet used and $b - a$ is greater than the previously constructed $\mathcal{P}$-position by one.

Theorem 8.28 (Beatty's Theorem[BACD+26])**.** *Let p and q be two positive reals numbers. Then the following assertions are equivalent.*

- *p and q are irrational and $\dfrac{1}{p} + \dfrac{1}{q} = 1$*
- *The two sequences, called Beatty sequences $(\lfloor np \rfloor)_{n\geq 1}$ and $(\lfloor nq \rfloor)_{n\geq 1}$ create a partition of $\mathbb{N}^+$.*

Moreover, $(n_k)_k$ and $(m_k)_k$ are the Beatty sequences associated with the equation:

$$\frac{1}{\Phi} + \frac{1}{\Phi^2} = 1$$

Thus, the two sequences $(n_k)_k$ and $(m_k)_k$ create a partition of $\mathbb{N}$.

Theorem 8.29. *For $k \geq 0$, $\kappa((n_k, m_k)) = 2k$.*

Proof. For t, an integer, from a t-$\mathcal{P}$-position the other t-$\mathcal{P}$-position cannot be reached, since they have the same total number of tokens. Thus, if both players play optimally, at most k other $\mathcal{P}$-positions will be encountered starting from (n_k, m_k). Hence, $\kappa((n_k, m_k)) \leq 2k$.

We consider the following strategy from a starting position. From (n_k, m_k), go to (n_{k-1}, m_k). Notice that $m_k - n_{k-1} > k$, and we know that for each $k' < k$, $m_{k'} - n_{k'} = k' < k$. Thus, the only accessible $\mathcal{P}$-position from (n_{k-1}, m_k) is the $k - 1^{\text{st}}$- $\mathcal{P}$-position (n_{k-1}, m_{k-1}). Hence, $\kappa((n_k, m_k)) \geq 2k$. □

8.4 . Difficulty in Computing the Sum of ISSGs

Contrary to the Grundy value of the sum of games that can be obtained from the Grundy value of the summed games, this is not the case for the κ value. More precisely, for two impartial games G and H, $\kappa(G + H)$ cannot be expressed just using $\kappa(G)$, $\kappa(H)$, $g(G)$ and $g(H)$. Even if we add more parameters κ^+ and κ^- defines as $\kappa^+(G)$ is the length of the game G when both players try to play optimally while slowing down the game as much as possible and $\kappa^-(G)$ the length of the game G when both players try

to play optimally while accelerating down the game as much as possible, then $\kappa(G+H)$ can still not be described as a function of $\kappa(G)$, $\kappa(H)$, $\kappa^+(G)$, $\kappa^+(H)$, $\kappa^-(G)$, $\kappa^-(H)$, $g(G)$ and $g(H)$.

In order to prove this limitation, we consider a game where adding edges does not change all those values.

Definition 8.30. *The accelerated bounded Nim game $\tilde{N}_k^{(m)}$ correspond to the game $N_k^{(m)}$ where we added the edges (i,j) for $j<i$, $g(i)g(j)\neq 0$ and $g(i)\neq g(j)$.*

In this game, the Grundy value of the positions are not changed and every play where both players play optimally need to pass through all the $\mathcal{P}$-positions reachable from k. Hence, we have the following equalities: $\kappa(\tilde{N}_k^{(m)}) = \kappa^+(\tilde{N}_k^{(m)}) = \kappa^-(\tilde{N}_k^{(m)}) = \kappa(N_k^{(m)}) = \kappa^+(N_k^{(m)}) = \kappa^-(N_k^{(m)})$. However, let consider the game $\tilde{N}_{99}^{(9)} + \tilde{N}_{99}^{(9)}$. We note that $\tilde{N}_{99}^{(9)} + \tilde{N}_{99}^{(9)}$ is a $\mathcal{P}$-position. Notice that if the first player chose to move in $\tilde{N}_{90+k}^{(9)} + \tilde{N}_{99}^{(9)}$, with $9>k>0$, then the second player can move to $\tilde{N}_{90+k}^{(9)} + \tilde{N}_{k}^{(9)}$ and the game will last at most 36 other moves if both players play accordingly to the slowed shortest time and the game will last at most 38 steps. If the first player start to go in $\tilde{N}_{90}^{(9)}$ then the game will last at most 40 moves. Thus $\kappa\left(\tilde{N}_{99}^{(9)} + \tilde{N}_{99}^{(9)}\right) = 40$. Notice that $\kappa\left(N_{99}^{(9)} + N_{99}^{(9)}\right) = 58$ by Theorem 8.24. Thus, in general,

$$\kappa(\tilde{N}_k^{(m)} + \tilde{N}_{k'}^{(m')}) \neq \kappa(N_k^{(m)} + N_{k'}^{(m')})$$

It results that the parameter previously listed are not enough to define the κ value of sums of games.

We recall that the κ value of impartial games defines the values of their skipping version. Thus, being able to sum ISSG solves the problem of computing the κ value of summed impartial games. Hence, summing ISSG is harder than computing κ.

Bibliography

[ABdMS21] David Auger, Xavier Badin de Montjoye, and Yann Strozecki. A Generic Strategy Improvement Method for Simple Stochastic Games. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:22, 2021.

[ACS14] David Auger, Pierre Coucheney, and Yann Strozecki. Finding optimal strategies of almost acyclic simple stochastic games. In *International Conference on Theory and Applications of Models of Computation*, pages 67–85, 2014.

[ACS19] David Auger, Pierre Coucheney, and Yann Strozecki. Solving Simple Stochastic Games with Few Random Nodes Faster Using Bland's Rule. In *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, pages 9:1–9:16, 2019.

[AHMS08] Daniel Andersson, Kristoffer Arnsfelt Hansen, Peter Bro Miltersen, and Troels Bjerre Sørensen. Deterministic graphical games revisited. In *Conference on Computability in Europe*, pages 1–10. Springer, 2008.

[AM09] Daniel Andersson and Peter Bro Miltersen. The complexity of solving stochastic games on graphs. In *International Symposium on Algorithms and Computation*, pages 112–121, 2009.

[BACD+26] Samuel Beatty, Nathan Altshiller-Court, Otto Dunkel, A. Pelletier, Frank Irwin, J. L. Riley, Philip Fitch, and D. M. Yost. Problems for solutions: 3173-3180. *The American Mathematical Monthly*, 33(3):159–159, 1926.

[BCG04] Elwyn R Berlekamp, John H Conway, and Richard K Guy. *Winning ways for your mathematical plays, volume 4*. AK Peters/CRC Press, 2004.

[BdM21] Xavier Badin de Montjoye. A recursive algorithm for solving simple stochastic games. *arXiv preprint arXiv:2110.01030*, 2021.

[Bel57] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[BMN+19] Luis Barba, Malte Milatz, Jerri Nummenpalo, Xiaoming Sun, Antonis Thomas, Jialin Zhang, and Zhijie Zhang. The complexity of optimization on grids. *Algorithmica*, 81(9):3494–3518, 2019.

[Bor21] Emile Borel. La théorie du jeu et les équations intégrales à noyau symétrique. *Comptes rendus de l'Académie des Sciences*, 173(1304-1308):58, 1921.

[BSV03] Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. Randomized subexponential algorithms for parity games. Technical Report 2003-019, Department of Information Technology, Uppsala University, April 2003.

[BSV04] Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. In *International Symposium on Mathematical Foundations of Computer Science*, pages 673–685. Springer, 2004.

[CdH13] Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. Strategy improvement for concurrent reachability and turn-based stochastic safety games. *Journal of Computer and System Sciences*, 79(5):640 – 657, 2013.

[CF11] Krishnendu Chatterjee and Nathanaël Fijalkow. A reduction from parity games to simple stochastic games. *arXiv preprint arXiv:1106.1232*, 2011.

[CFK+13] Taolue Chen, Vojtěch Forejt, Marta Kwiatkowska, David Parker, and Aistis Simaitis. Automatic verification of competitive stochastic systems. *Formal Methods in System Design*, 43(1):61–92, 2013.

[CJK+20] Cristian S Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasi-polynomial time. *SIAM Journal on Computing*, pages STOC17–152, 2020.

[CK78] Seth Chaiken and Daniel J Kleitman. Matrix tree theorems. *Journal of combinatorial theory, Series A*, 24(3):377–381, 1978.

[CKS81] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *JACM*, 28(1), 1981.

[CKSW13] Taolue Chen, Marta Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche. Synthesis for multi-objective stochastic games: An application to autonomous urban driving. In *International Conference on Quantitative Evaluation of Systems*, pages 322–337. Springer, 2013.

[Con90] Anne Condon. On algorithms for simple stochastic games. pages 51–72, 1990.

[Con92] Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

[Con93] Anne Condon. On algorithms for simple stochastic games. *Advances in computational complexity theory*, 13:51–73, 1993.

[Con00] John H Conway. *On numbers and games*. AK Peters/CRC Press, 2000.

[DG09] Decheng Dai and Rong Ge. New results on simple stochastic games. In *International Symposium on Algorithms and Computation*, pages 1014–1023. Springer, 2009.

[DTW03] Ioana Dumitriu, Prasad Tetali, and Peter Winkler. On playing golf with two balls. *SIAM Journal on Discrete Mathematics*, 16(4):604–615, 2003.

[EJ91] E Allen Emerson and Charanjit S Jutla. Tree automata, mu-calculus and determinacy. In *FoCS*, volume 91, pages 368–377. Citeseer, 1991.

[EJS93] E Allen Emerson, Charanjit S Jutla, and A Prasad Sistla. On model-checking for fragments of μ-calculus. In *International Conference on Computer Aided Verification*, pages 385–396. Springer, 1993.

[EM79] Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8(2):109–113, 1979.

[FBB+21] Nathanaël Fijalkow, Nathalie Bertrand, Romain Brenguier, Patricia Bouyer-Decitre, Arnaud Carayol, John Fearnley, Hugo Gimbert, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, Benjamin Monmege, Petr Novotny, Mickael Randour, Ocan Sankur, Sylvain Schmitz, and Serre Olivier. *Games on Graphs*. to appear, 2021.

[Fea10] John Fearnley. Exponential lower bounds for policy iteration. In *International Colloquium on Automata, Languages, and Programming*, pages 551–562. Springer, 2010.

[FGMS20] John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique end of potential line. *Journal of Computer and System Sciences*, 114:1–35, 2020.

[Fri09] Oliver Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. In *2009 24th Annual IEEE Symposium on Logic In Computer Science*, pages 145–156. IEEE, 2009.

[Fri11] Oliver Friedmann. An exponential lower bound for the latest deterministic strategy iteration algorithms. *Logical Methods in Computer Science*, 7, 2011.

[G+03] Branko Gr et al. *Convex Polytopes*, volume 221. Springer Science & Business Media, 2003.

[GH08] Hugo Gimbert and Florian Horn. Simple stochastic games with few random vertices are easy to solve. In *Foundations of Software Science and Computational Structures*, pages 5–19. Springer, 2008.

[GH09] Hugo Gimbert and Florian Horn. Solving simple stochastic games with few random vertices. *Logical Methods in Computer Science*, Volume 5, Issue 2, 2009.

[GJMR08] Bernd Gärtner, Walter D Jr Morris, and Leo Rüst. Unique sink orientations of grids. *Algorithmica*, 51(2):200–235, 2008.

[GR63] G Th Guilbaud and Pierre Rosenstiehl. Analyse algébrique d'un scrutin. *Mathématiques et Sciences humaines*, 4:9–33, 1963.

[GR01] Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2001.

[Gru39] Patrick M Grundy. Mathematics and games. *Eureka*, 2:6–9, 1939.

[GT02] Erich Gradel and Wolfgang Thomas. Automata, logics, and infinite games: a guide to current research. 2002.

[Hal07] Nir Halman. Simple stochastic games, parity games, mean payoff games and discounted payoff games are all lp-type problems. *Algorithmica*, 49(1):37–50, 2007.

[HK66] Alan J Hoffman and Richard M Karp. On nonterminating stochastic games. *Management Science*, 12(5):359–370, 1966.

[HMZ13] Thomas Dueholm Hansen, Peter Bro Miltersen, and Uri Zwick. The complexity of solving stochastic games on graphs. *Journal of the ACM (JACM)*, 60(1):1–16, 2013.

[How60] Ronald A Howard. Dynamic programming and markov processes. 1960.

[IJM12] Rasmus Ibsen-Jensen and Peter Bro Miltersen. Solving simple stochastic games with few coin toss positions. In *European Symposium on Algorithms*, pages 636–647. Springer, 2012.

[JSWZ20] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. *arXiv preprint arXiv:2004.07470*, 2020.

[Jub05] Brendan Juba. On the hardness of simple stochastic games. *Master's thesis, CMU*, 2005.

[KM03] Stephen Kwek and Kurt Mehlhorn. Optimal search for rationals. *Information Processing Letters*, 86(1):23–26, 2003.

[Knu11] Donald E Knuth. *The art of computer programming, volume 4A: combinatorial algorithms, part 1*. Pearson Education India, 2011.

[KRSW22] Jan Křetínskỳ, Emanuel Ramneantu, Alexander Slivinskiy, and Maximilian Weininger. Comparison of algorithms for simple stochastic games. *Information and Computation*, page 104885, 2022.

[LR04] Gilat Levy and Ronny Razin. It takes two: an explanation for the democratic peace. *Journal of the European economic Association*, 2(1):1–29, 2004.

[Lud95] Walter Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and computation*, 117(1):151–155, 1995.

[MC90] Mary Melekopoglou and Anne Condon. *On the complexity of the policy iteration algorithm*. University of Wisconsin-Madison, Computer Sciences Department, 1990.

[Mos91] Andrzej Włodzimierz Mostowski. *Games with forbidden positions*. 1991.

[MSW96] Jiří Matoušek, Micha Sharir, and Emo Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4):498–516, 1996.

[Ovc11] Sergei Ovchinnikov. *Graphs and cubes*. Springer Science & Business Media, 2011.

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

[Pap94] Christos H Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and system Sciences*, 48(3):498–532, 1994.

[PTVF07] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[Pur97] Anuj Puri. Theory of hybrid systems and discrete event systems. 1997.

[Que53] Adolphe Quetelet. *Théorie des probabilités*. Soc. lémancipation intellectuelle, 1853.

[Rab63] Michael O Rabin. Probabilistic automata. *Information and control*, 6(3):230–245, 1963.

[Sha53] L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953.

[Sko21] Mateusz Skomra. Optimal bounds for bit-sizes of stationary distributions in finite markov chains. *arXiv preprint arXiv:2109.04976*, 2021.

[Som05] Rafał Somla. New algorithms for solving simple stochastic games. *Electronic Notes in Theoretical Computer Science*, 119(1):51–65, 2005.

[SP73] JMPGR Smith and George R Price. The logic of animal conflict. *Nature*, 246(5427):15–18, 1973.

[SPR35] R. SPRAGUE. Über mathematische kampfspiele. *Tohoku Mathematical Journal, First Series*, 41:438–444, 1935.

[Sti99] C Stirling. Bisimulation, modal logic and model checking games. *Logic Journal of the IGPL*, 7(1):103–124, 1999.

[SW78] Alan Stickney and Layne Watson. Digraph models of bard-type algorithms for the linear complementarity problem. *Mathematics of Operations Research*, 3(4):322–333, 1978.

[SW01] Tibor Szabó and Emo Welzl. Unique sink orientations of cubes. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 547–555. IEEE, 2001.

[Tho06] Rekha R Thomas. *Lectures in geometric combinatorics*, volume 33. American Mathematical Soc., 2006.

[TVK11] Rahul Tripathi, Elena Valkanova, and VS Anil Kumar. On strategy improvement algorithms for simple stochastic games. *Journal of Discrete Algorithms*, 9(3):263–278, 2011.

[VNM07] John Von Neumann and Oskar Morgenstern. Theory of games and economic behavior. In *Theory of games and economic behavior*. Princeton university press, 2007.

[Wyt07] Willem A Wythoff. A modification of the game of nim. *Nieuw Arch. Wisk*, 7(2):199–202, 1907.

[Zer13] Ernst Zermelo. Über eine anwendung der mengenlehre auf die theorie des schachspiels. In *Proceedings of the fifth international congress of mathematicians*, volume 2, pages 501–504. Cambridge University Press Cambridge, 1913.

[ZP96] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996.