Games with incomplete information : complexity, algorithmics, reasoning

Junkang Li

Normandie Université

UNIVERSITÉ CAEN NORMANDIE

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité **INFORMATIQUE**

Préparée au sein de l'**Université de Caen Normandie**

Jeux à information incomplète : complexité, algorithmique, raisonnement.

Présentée et soutenue par

JUNKANG LI

Thèse soutenue le 21/12/2023

devant le jury composé de :

M. OLIVIER SPANJAARD	Maître de conférences HDR - UNIVERSITE PARIS 4 PARIS-SORBONNE	Rapporteur du jury
M. NATHAN STURTEVANT	Professeur - University of Alberta	Rapporteur du jury
M. TRISTAN CAZENAVE	Professeur des universités - UNIVERSITE PARIS 9	Membre du jury
M. VINCENT THOMAS	Maître de conférences - Université de Lorraine	Membre du jury
MME VERONIQUE VENTOS	Docteur - NUKKAI	Membre du jury
M. CATALIN DIMA	Professeur des universités - UNIVERSITE MARNE LA VALLEE UNIV PARIS EST MARNE LA VALLEE	Président du jury
M. BRUNO ZANUTTINI	Professeur des universités - Université de Caen Normandie	Directeur de thèse

Thèse dirigée par **BRUNO ZANUTTINI** (GREYC ALGORITHMIQUE)

MIIS

GREYC
Laboratoire de recherche en sciences du numérique

Games with incomplete information: complexity, algorithmics, reasoning

A study inspired by the game of Bridge

LI Junkang

Advisors: Bruno Zanuttini, Véronique Ventos

Dedication

LI Danlin,
uxori deliciisque carissimis meis,
hoc opus dedicandum est.
Nisi eius amore sustentus essem,
hoc perficere non potuissem.

Contents

Résumé

Introduction

Théorie des jeux

La *théorie des jeux*, qui propose une modélisation des interactions entre agents (aussi appelés joueurs), est un domaine interdisciplinaire qui se situe à l'intersection entre l'économie, l'informatique et les mathématiques. En particulier, c'est une branche importante de l'intelligence artificielle.

La théorie des jeux se scinde principalement en deux branches : la première concerne les jeux coopératifs (modélisés par la formation de coalition d'agents), tandis que la seconde, à laquelle on se consacre dans ce travail, étudie les jeux non coopératifs. Les jeux non coopératifs sont surtout représentés sous deux formes : la forme *normale* et la forme *extensive*.

- Sous la forme normale, un jeu est défini par un ensemble de joueurs, un ensemble de stratégies possibles pour chaque joueur, et une fonction d'utilité qui associe une récompense, pour chaque joueur, à chaque combinaison possible des stratégies possibles des joueurs. Intuitivement, un tel jeu décrit une situation dans laquelle les joueurs interagissent en un tour de façon simultanée : chacun choisit simultanément sa stratégie dans son ensemble de stratégies possibles, puis reçoit une récompense qui est uniquement déterminée par la fonction d'utilité, en fonction des stratégies choisies par les joueurs.

- Sous la forme extensive, un jeu est défini par un arbre muni d'informations supplémentaires, comme le propriétaire d'un nœud interne ou les récompenses des joueurs à une feuille. Un tel jeu commence à la racine de son arbre et procède de cette manière : pour chaque nœud interne, son propriétaire doit choisir un enfant/successeur de ce nœud. Le jeu est terminé quand une feuille est atteinte, et chaque joueur reçoit alors une récompense, qui dépend de la feuille atteinte. Dans un jeu sous forme extensive, une stratégie d'un joueur est une application qui associe chaque nœud qui lui appartient à un enfant de ce nœud.

La forme normale et la forme extensive sont deux représentations équivalentes, en ce sens qu'un jeu exprimé sous une forme peut être exprimé sous l'autre. Cependant, la taille de la forme normale d'un jeu est souvent exponentielle en la taille de sa forme extensive. De surcroît, représenter un jeu sous forme normale masque l'aspect dynamique de ce jeu. Ainsi, on se permet de se concentrer sur les jeux sous forme extensive, d'autant plus l'objectif de ce travail est de modéliser les jeux de table auxquels les humains jouent.

Catégories de jeux sous forme extensive

Il est possible de modéliser explicitement les facteurs de chance dans un jeu sous forme extensive, en attribuant certains nœuds internes à un joueur spécial dénommé *Nature*. À chacun de ses nœuds (nommés *nœuds de hasard*), Nature choisira un successeur de façon probabiliste, selon une distribution de probabilités connue de tous les joueurs du jeu. Selon l'existence ou non de nœuds de hasard, un jeu est appelé un *jeu de hasard* ou un *jeu sans hasard*.

Parfois, un joueur dans un jeu ne possède pas une information parfaite : par exemple, il ne connaît pas tous les choix effectués par les joueurs dans le passé. Autrement dit, ce joueur peut ignorer où il est dans l'arbre quand il doit prendre une décision. Une telle situation peut être modélisée en partitionnant les nœuds d'un joueur en des *ensembles d'information*, et en imposant que ce joueur doive prendre la même action à tous les nœuds dans le même ensemble d'information. Intuitivement, pendant le déroulement d'un tel jeu, quand un joueur doit choisir une action, il est seulement informé de l'ensemble d'information auquel le nœud courant appartient, mais pas du nœud lui-même. Si les ensembles d'information d'un joueur sont tous des singletons, on dit que ce joueur a une *information parfaite* ; sinon, il a une *information imparfaite*. Strictement entre ces deux notions d'information se situe la notion nommée *mémoire parfaite* (*perfect recall*), qui signifie intuitivement qu'un joueur se souvient toujours des informations qu'il a reçues. En fonction du degré d'information des joueurs, on peut classer les jeux sous forme extensive en différentes catégories.

Enfin, la notion d'*information incomplète* modélise les situations dans lesquelles un joueur ne connaît pas tous les éléments concernant le jeu auquel il joue. Par exemple, il peut ne pas connaître le nombre de joueurs dans le jeu, les actions disponibles ou les gains d'autres joueurs, etc. Bien que les jeux à information incomplète puissent être modélisés comme des jeux à information imparfaite, ils forment eux-mêmes une catégorie importante de jeux sous forme extensive.

Concepts de solution

La théorie des jeux est une théorie *descriptive* et non *normative* : elle vise à *prédire*, non pas à *prescrire* le déroulement d'un jeu. Plus concrètement, il s'agit de la notion de *concept de solution*. Chaque concept de solution impose un ensemble de contraintes, afin de limiter l'attention que l'on porte sur un jeu aux dénouements *raisonnables* de ce jeu. Par exemple, le fameux concept de solution de l'*équilibre de Nash* juge que les solutions raisonnables d'un jeu sont celles dans lesquelles personne n'a intérêt à dévier sa stratégie de façon unilatérale. Dans notre travail, on s'intéresse particulièrement au concept de solution nommé *maxmin*, qui prédit que chaque joueur choisit une stratégie qui vise à maximiser son gain minimum (lorsqu'il considère toutes les stratégies adverses).

Cadre du travail

Inspiré fortement par le bridge, ce travail consiste à étudier les jeux à information incomplète à deux joueurs et le calcul lié au concept de solution maxmin pour ces jeux, sous différents angles : complexité, algorithmique, raisonnement.

Complexité On commencera par une étude complète de la complexité du calcul de la valeur maxmin pour différentes catégories de jeux. Les résultats permettront de

bien cerner les facteurs qui augmentent la complexité de résoudre un jeu ; ils assurent également que certaines hypothèses simplificatrices que l'on fera dans la suite ne conduiront pas à une perte de généralité.

Algorithmique Ensuite, on se consacrera à l'étude des algorithmes et de leurs optimisations pour les jeux à information incomplète comme le bridge.

Raisonnement Finalement, on donnera un rapide aperçu sur les raisonnements d'un joueur sur les stratégies d'autres joueurs que l'on observe très souvent dans les jeux à information incomplète, notamment le bridge. On proposera un formalisme récursif pour modéliser ces raisonnements, et des algorithmes pour implémenter ce formalisme.

Dans notre travail, on se concentre sur la complexité et les algorithmes autour de la valeur *maxmin pure*, qui est le gain qu'un joueur peut garantir en jouant l'une de ses stratégies possibles. Bien entendu, on peut aussi autoriser un joueur à utiliser le hasard dans sa stratégie. Par exemple, il peut faire un tirage au sort à chaque nœud qui lui appartient pour décider l'action qu'il choisira, comme ce que fait Nature, ce qui correspond à la notion de *stratégie comportementale*. Il peut également faire un tirage au sort avant le début du jeu pour choisir une stratégie possible qu'il suivra tout au long du jeu, ce qui correspond à la notion de *stratégie mixte*.

Le concept de solution est bien défini par rapport à ces deux notions de stratégies. En revanche, on s'intéresse principalement à la valeur maxmin pure, plutôt qu'à la valeur maxmin comportementale ou mixte, pour les raisons suivantes.

- Le calcul de la valeur maxmin comportementale ou mixte est très bien étudié dans la littérature : sa complexité est connue, et il y a de nombreux travaux sur les algorithmes pour les calculer exactement ou approximativement.

- Depuis longtemps, il a été constaté que les humains ont tendance à éviter les stratégies probabilistes (comportementales ou mixtes) et à raisonner en termes de stratégies pures. En effet, il y a un domaine entier, appelé *théorie des jeux comportementale*, qui étudie ce genre de phénomènes, et plus largement la divergence entre ce que prédisent les concepts de solution classiques et ce que réellement font les humains. Ainsi, pour une intelligence artificielle qui devra collaborer ou se placer en concurrence avec des humains, il est primordial d'être capable de raisonner en termes des stratégies pures, ce qui justifie le cadre de ce travail.

Dans la suite, on résume les résultats par chapitre.

Complexité des jeux sous forme extensive

Complexité des EFGs

On commence ce travail par une étude sur la complexité de trouver une borne inférieure de la valeur maxmin pure pour les jeux sous forme extensive (*extensive-form games*, ou EFGs ci-après) ; on nomme le problème de décision associé PURE MAXMIN.

Les résultats concernant la valeur maxmin comportementale ou mixte (c'est-à-dire BEHAVIOUR MAXMIN et MIXED MAXMIN) sont déjà connus dans la littérature, que l'on présente dans la table 1 pour faciliter la comparaison avec nos résultats. Dans ce

tableau, les rangées correspondent aux différents degrés d'information pour le joueur MAX (le joueur dont on cherche à maximiser le gain) : IP pour information parfaite, MP pour mémoire parfaite, MP-MA pour mémoire parfaite multi-agents (c'est-à-dire que MAX est une équipe de plusieurs agents partageant le même gain, et ayant chacun une mémoire parfaite) ; les colonnes correspondent aux différents degrés d'information pour le joueur MIN, l'unique opposant de MAX. Dans ce tableau, la complexité est croissante en les deux dimensions ; on constate aussi que le fait d'être multi-agents rend un jeu plus difficile à résoudre, ce qui est conforme à notre intuition.

Notons que la table 1 concerne la complexité des EFGs avec hasard. Pour les EFGs sans hasard, leur complexité coïncide avec celle des EFGs avec hasard, sauf dans un cas encore ouvert qui concerne les EFGs avec mémoire parfaite multi-agents ; on pourra montrer que même dans ce cas, avec ou sans hasard ne change pas la complexité.

MAX \ MIN	IP	MP	MP-MA
IP	P	P	coNP-c
MP	P	P	coNP-c
MP-MA	NP-c	NP-c	Σ_2^P-c/Δ_2^P-c

Table 1: Complexité de Behaviour Maxmin et de Mixed Maxmin pour les EFGs avec hasard. Dans la dernière cellule, Behaviour Maxmin et Mixed Maxmin sont respectivement Σ_2^P-complet et Δ_2^P-complet.

Les résultats de Pure Maxmin sont présentés dans la table 2 ; ceux en caractères gras sont nouveaux. Pour montrer ces résultats, on propose un nouvel algorithme en temps polynomial. On utilise aussi des réductions minimales, en ce sens qu'elles impliquent au plus deux agents de chaque équipe (MAX et MIN), et seulement des récompenses Booléennes. Quand on contraste la table 2 avec la table 1, on voit que le paysage de complexité est très différent si MAX peut uniquement jouer des stratégies pures :

- dans la table 2, la présence de hasard change complètement la complexité dans beaucoup de cas, parfois même de P à Σ_2^P, ce qui n'est pas le cas dans la table 1 ;
- en l'absence de hasard, le degré d'information de MIN ne joue pas dans la complexité de Pure Maxmin, contrairement à Behaviour Maxmin ou à Mixed Maxmin ;
- parfois, Pure Maxmin est plus difficile que Behaviour Maxmin et Mixed Maxmin ; parfois, c'est le contraire.

	Sans hasard	Avec hasard		
MAX \ MIN	PI/MP/MP-MA	PI	MP	MP-MA
PI	P	P	**NP-c**	$\mathbf{\Sigma_2^P}$**-c**
MP	**P**	NP-c	NP-c	$\mathbf{\Sigma_2^P}$**-c**
MP-MA	NP-c	NP-c	NP-c	Σ_2^P-c

Table 2: Complexité de Pure Maxmin.

Complexité des jeux représentés de façon compacte

On étudie ensuite la complexité des jeux représentés de façon compacte. La motivation est naturelle : les jeux de table auxquels on joue sont rarement définis explicitement par leur arbre de jeu, mais le sont plutôt par leurs *règles* et par quelques paramètres décrivant la taille du jeu. Par exemple, le jeu de go est défini par ses règles et par la taille du goban (traditionnellement de taille 19×19, mais possiblement d'autres dimensions) ; le jeu du bridge est défini par ses règles et par le nombre de rangs et de couleurs (normalement 4 couleurs, chacune avec 13 rangs). Ainsi, pour ces jeux, ce sont ces paramètres, plutôt que la taille de leur arbre de jeu, qui sont pertinents pour analyser la complexité.

Pour modéliser les jeux représentés de façon compacte, on propose deux formalismes diamétralement opposés.

- Le premier, que l'on nomme *jeu Booléen compact* (CBG ci-après), est une généralisation des formules Booléennes quantifiées (avec ou sans dépendances ; QBF ci-après), qui sont bien connues dans la littérature, et encodent des jeux sous forme extensive de façon compacte. On propose le formalisme CBG pour généraliser de façon minimale QBF, tout en autorisant des facteurs de hasard et des équipes multi-agents.

- En revanche, le second, que l'on nomme *jeu avec oracle* (OG ci-après), est très générique, et permet de capturer les jeux qui se jouent en espace polynomial et avec un horizon polynomial, ce qui correspond à la plupart des jeux de table.

On peut vérifier facilement qu'OG est effectivement plus expressif que CBG. On montre ensuite les résultats de difficulté (pour une classe de complexité) pour CBG, et les résultats d'appartenance pour OG, ce qui nous permet de confirmer :

- que ces deux formalismes (et donc également tous les formalismes intermédiaires) sont équivalents en termes de complexité (à des transformations polynomiales près) ;

- que la notion de complexité utilisée est robuste au formalisme utilisé pour les jeux représentés de façon compacte, et donc qu'elle représente bien la difficulté intrinsèque de ces jeux, indépendamment de la représentation choisie.

Sans surprise, la complexité augmente de manière exponentielle quand un jeu est défini de façon compacte plutôt que par son arbre de jeu. Concrètement, les problèmes inclus dans P dans la table 2 deviennent PSPACE-complets ; ceux qui sont NP-complets deviennent NEXP-complets ; ceux qui sont Σ_2^{P}-complets deviennent $\mathsf{NEXP}^{\mathsf{NP}}$-complets.

Autres résultats de complexité

Enfin, on étudie aussi la complexité de calculer les stratégies optimales de MAX quand MIN choisit sa stratégie dans un ensemble restreint et connu par MAX. Ces stratégies de MIN sont appelées *modèles d'opposants* (OM ci-après) dans la littérature ; cette notion d'OM permet de simuler les situations dans lesquelles MAX connaît (parfaitement ou partiellement) la procédure de raisonnement de MIN. Les résultats sont présentés dans la table 3, où les colonnes correspondent au nombre d'OM connus par MAX. Dans chaque cellule, il y a deux cases : celle du haut correspond aux EFGs sans hasard, et celle du bas aux EFGs avec hasard.

Hazard \ $\lvert\Sigma^O_-\rvert$ / MAX	1	constant (≥ 2)	non borné
PI	P	P	P
	P	**NP-c**	**NP-c**
PR	P	P	**P**
	P	**NP-c**	NP-c
MA-PR	P	**P**	NP-c
	NP-c	NP-c	NP-c

Table 3: Complexité de PURE OM-MAXMIN.

Enfin, on étudie la complexité des problèmes liés à une borne supérieure ou à la valeur exacte de la valeur maxmin pure. Il s'avère que la complexité de ces problèmes est intimement liée aux résultats de la table 2.

Jeux combinatoires à information incomplète

On se focalise ensuite sur les jeux à information incomplète. Motivé par le bridge, on propose un nouveau formalisme, nommé *jeu combinatoire à information incomplète* (CGII ci-après).

Un CGII est défini par un arbre de jeu *public*, un modèle d'Aumann de l'information incomplète (c'est-à-dire un modèle de Kripke S5), et une fonction de récompense. Au début d'un tel jeu, Nature tire un monde dans un univers fixé et connu par tous les joueurs ; chaque joueur observe partiellement ou parfaitement ce monde. Par exemple, pour un jeu de carte comme le bridge, l'univers comprend toutes les distributions possibles des cartes aux joueurs ; chaque joueur observe sa propre main, mais pas la main des autres joueurs. Ensuite, le jeu continue dans son arbre public comme dans un EFG sans hasard, dans lequel les actions choisies par les joueurs sont parfaitement observables par tous. À la fin, la récompense d'un joueur peut dépendre à la fois de la feuille atteinte et du monde choisi par Nature.

En plus d'être motivé par le souhait de modéliser le bridge, ce nouveau formalisme est conçu délibérément pour :

- généraliser de façon minimale les jeux combinatoires (c'est-à-dire les jeux à information parfaite et sans hasard) pour inclure aussi la possibilité qu'un joueur ait une information incomplète ;

- par rapport aux EFGs, mieux capturer l'essence de l'information incomplète due au fait que toutes les actions sont publiques : en effet, dans un CGII, l'information imparfaite d'un joueur pendant le jeu vient seulement de son observation partielle du monde tiré par Nature (qui est aussi la source de l'information incomplète), et non des actions cachées effectuées par les autres joueurs.

Au premier regard, cette définition de CGII paraît excessive, car l'arbre est public et sans nœud de hasard. Pourtant, ces restrictions sont bien motivées et justifiées. Par exemple, les facteurs de hasard peuvent être encodés par le tirage initial d'un monde ; une application remarquable de cette idée est la génération procédurale dans les jeux vidéos, dont la sortie est complètement déterminée par une unique graine aléatoire.

L'argument le plus fort favorisant ce nouveau formalisme est donné par les résultats de complexité de PURE MAXMIN pour les CGIIs. Certes, les constructions sont beaucoup plus complexes, du fait de l'absence d'actions cachées dans les CGIIs, mais les résultats de complexité sont exactement les mêmes que ceux pour les EFGs. On en déduit que le formalisme des CGIIs, en dépit de sa simplicité, est aussi expressif (à un facteur polynomial près) que le formalisme des EFGs.

Algorithmes et optimisations pour les CGIIs

On poursuit par une étude algorithmique, en faisant l'hypothèse simplificatrice que MAX a une information incomplète tandis que MIN a une information complète. On appelle les CGIIs satisfaisant cette propriété les *jeux vectoriels*, car les récompenses de ces jeux peuvent être représentées comme des vecteurs de réels.

Le modèle de la meilleure défense permet de transformer tous les CGIIs en des CGIIs dans lesquels MIN a une information complète. La valeur maxmin d'un CGII est toujours bornée inférieurement par celle du jeu obtenu via cette transformation ; empiriquement, pour le bridge, les deux valeurs sont souvent très proches, voire égales. L'intérêt principal de cette transformation est que le jeu transformé est en général beaucoup plus facile à résoudre que le jeu initial, ce que les résultats de complexité confirment.

On montre d'abord les difficultés particulières du calcul de la valeur maxmin pure d'un CGII (notamment la fusion de stratégies et la non-localité). Ensuite, on présente un algorithme en recherche par profondeur, initialement conçu sans preuve par Ginsberg en 2001, qui calcule la valeur maxmin pure exacte.

Vu que le problème sous-jacent de l'algorithme de Ginsberg est NP-dur, cet algorithme n'est naturellement pas en temps polynomial. Plus précisément, il tourne en temps linéaire en la taille de l'arbre, mais exponentiel en la taille de l'univers (le paramètre qui quantifie le degré d'information incomplète du joueur MAX). En conséquence, des optimisations sont indispensables afin de le faire passer à l'échelle. On regroupe les optimisations que l'on étudie dans notre travail en deux catégories : élagage de stratégies et élagage de l'arbre de jeu.

L'exemple principal d'élagage de stratégies est donné par l'élimination des stratégies dominées. On formule une méthode d'élagage de stratégies par un objet appelé fonction de choix (venant du domaine de la théorie du choix social). On présente des critères quasiment nécessaires et suffisants afin qu'une fonction de choix soit une méthode d'élagage correcte (c'est-à-dire qu'elle ne change pas la valeur maxmin calculée). On montre aussi que sous ces mêmes conditions, appliquer partiellement un élagage partout dans l'arbre reste un élagage correct.

Quant à l'élagage de l'arbre de jeu, on étend l'élagage alpha-bêta à un cadre encore plus général que celui des jeux vectoriels, qui est l'évaluation d'un graphe et/ou avec des valeurs prises dans un treillis. On montre que l'élagage alpha-bêta reste correct dans ce cadre, même quand on initialise la valeur d'un nœud lors de la recherche arborescente, pourvu que la valeur d'initialisation satisfasse une certaine condition d'admissibilité. Un bonus de ce résultat est une caractérisation intuitive de la valeur renvoyée par une recherche alpha-bêta, en fonction de la valeur exacte et de la fenêtre de recherche (c'est-à-dire la fenêtre définie par alpha et bêta). On conçoit aussi une manière de munir l'élagage alpha-bêta sous ordre partiel d'un cache. Les expériences sur des *benchmarks* synthétiques montrent la performance de nos algorithmes par rapport à un algorithme minimax simple ou une recherche alpha-bêta sans cache.

Enfin, on ajoute un troisième paramètre, noté gamma, à l'élagage, pour obtenir un nouvel algorithme que l'on nomme l'élagage alpha-bêta-gamma. Ce nouveau paramètre gamma n'est pas mis à jour de façon mécanique comme alpha ou bêta lors de la recherche ; il permet donc d'encoder des heuristiques (par exemple, celles venant des connaissances d'un expert humain) pour mieux guider la recherche. Cet algorithme capture aussi des méthodes de recherche arborescente (nommées élagage CI et élagage PWS dans notre travail) utilisées par les joueurs humains de bridge.

Raisonnement récursif et modèles d'opposants

Dans le dernier chapitre, qui représente un travail en cours, on parle des motivations et des pistes possibles pour aller au-delà du modèle de la meilleure défense, qui surestime le pouvoir de MIN en supposant qu'il a une information complète. L'approche la plus élégante et prometteuse pour cet objectif semble être l'idée du raisonnement récursif. Ce type de raisonnement capture notamment certains algorithmes existants pour résoudre des jeux (*fictitious play*, l'algorithme de double oracle, la dynamique de meilleure réponse, etc.), mais également des modèles récursifs proposés par la théorie des jeux comportementale pour modéliser les comportements des humains.

Pour implémenter un tel raisonnement récursif dans le cadre des jeux vectoriels, il faut être capable de trouver des meilleures réponses en présence d'OM (calculés au niveau précédent de la récursion). Ainsi, on se ramène à étudier différents types d'OM : probabiliste, lexicographique, non déterministe, etc. Pour chaque type d'OM que l'on considère, on propose un algorithme de recherche en profondeur pour calculer une meilleure réponse contre les OM ; ces algorithmes sont présentés sous une forme unifiée, et ils sont linéaires en temps quand le problème sous-jacent n'est pas NP-complet. Pour conclure, on montre comment ces algorithmes s'utilisent dans un raisonnement récursif, en étudiant un exemple venant d'une donne de bridge jouée dans une compétition.

Acknowledgements

Academic-wise

I would like to thank Bruno Zanuttini, my advisor at university. It has been a great pleasure to work with you on the fascinating problems in this dissertation. Apart from your precious academic guidance, I especially want to thank you for the copious time you have dedicated to working with me or to reading scrupulously all my writings, despite being encumbered by multiple other responsibilities.

Next, I would like to thank Véronique Ventos, my supervisor at NukkAI. Without our serendipitous encounter, your vision behind the creation of NukkAI, or your enthusiastic support, I would not have switched from theoretical physics to computer science, nor embarked on the journey leading to this dissertation. Also, thank you for giving me carte blanche to choose my research subjects throughout my PhD, for which I am particularly grateful.

I would also like to thank Nathan Sturtevant and Olivier Spanjaard for accepting to be the reviewers of my dissertation; hopefully, it entertains you.

Work-wise

It goes without saying that I am indebted to my colleagues at NukkAI, including (but not limited to) our office manager David Touitou (thank you for taking care of all the administrative details of and beyond my PhD, which can sometimes be daunting for a foreigner in France); our CTO Swann Legras (thank you for all the technical support); our consultant Tristan Cazenave (thank you for your academic input to my research and for accepting to be in the jury of my defence); our project manager Jean-Pierre Desmoulins (thank you for being a role model in terms of work-related organisation and for having directed the writing of *Le bridge français*[1]); our close friend and world champion of Bridge Philippe Cronier (thank you for providing me with materials on psychological strategies in Bridge). I thank all my colleagues for creating a convivial atmosphere to work in. Special thanks go to our CEO, Jean-Baptiste Fantun, for his vision resulting in co-founding NukkAI with Véronique Ventos, and for his support and trust in my work.

Life-wise

First and foremost, I want to thank my wife LI Danlin. As stated in the dedication, I would not have completed my PhD without your love and support, for 11 years and counting. I truly hope that this dissertation bodes well for our future. I am also grateful

[1] This is the three-volume introduction that finally got me into Bridge, the complex game that has inspired this work; hence it worths a special mention.

to my parents, LI Yaoming and YU Hongmei, for their endless encouragement and love throughout my life; this gratitude also extends to my maternal grandparents, especially to my grandfather, who piqued my curiosity in science when I was so little and who encouraged me to go aboard for pursing my study in fundamental sciences. I also owe much to my teachers from *classe préparatoire aux grandes écoles* at IFCEN for my scientific (and linguistic) training. Lastly, I express my gratitude to all my friends, in particular my classmates from class 12 of my junior high school, who have had great influence on my personal development and passions (e.g. my predilection for games with incomplete information).

Caveat

Despite considerable efforts, this list of acknowledgements can never be exhaustive. So, a final thank you to everyone, mentioned above or not, that has been kind to me and conducive to the fruition of this dissertation.

Part I

Introduction and background

Chapter 1

Introduction

General presentation

Inspired and motivated by the card play of the game of Bridge,[1] this dissertation concerns games with incomplete information, from the perspective of computational complexity, algorithmics, and reasoning.

Game theory is a mathematical framework for studying multi-agent interactions, in which each agent chooses their action independently and the outcome (and payoffs) depends on the action chosen by all the agents. We focus on extensive-form games, in which the interaction between agents takes place sequentially, i.e. every agent takes turns to make a move. Prominent examples of such games are Chess and Go. Of particular interest to us is the notion of games with incomplete information, which are games in which agents do not have common knowledge of the game they play. For instance, an agent does not know the number of participants in an auction, or how much these participants value the object to be sold; a Poker player does not see the cards in their opponent's hidden hands, hence cannot know for sure the exact consequence (i.e. payoff) of calling and raising bets; a Bridge or Hearts player does not know the cards that their opponent can play during a trick since this depends on their hidden hand; etc. We are interested in knowing how much reward an agent can guarantee for themselves in these games; this corresponds to the notion of maxmin value, well known in optimisation under uncertainty, in which we aim to ensure that the worst possible outcome is not too bad. The choice to study this solution concept is initially motivated by the card play phase of Bridge, in which declarer searches for optimal and robust strategies that maximise their winning chance (with respect to hidden hands of defenders and against all strategies of defenders), which is exactly captured by the notion of maxmin value.

We study the computational complexity of games, which characterises how hard it is to play well in games. Intuitively, games involving hidden information or more players are more difficult, which is indeed confirmed by the study of complexity. Such results are also indicators of how unlikely we are to find efficient and general algorithms to solve "all" games. The more an algorithm is adapted to a particular problem, the less general it is, but the more efficient it can be by exploiting the features of the problem. This intrinsic trade-off between generality and efficiency needs to be taken into account when we design algorithms to tackle real-life problems.

[1] The rules of Bridge can be found in Appendix A.4.

Reasoning in games is also a fascinating topic. When humans interact with each other, in games as well as in other social contexts, they constantly perform such reasoning, which comes in many different forms. For example, we can have epistemic reasoning, about knowledge (I have this card in my hand, and I know that my opponent knows that it is in my hand, but they do not know that I know that they know; hence I can trick them in this way[2]); probabilistic reasoning, about events (if it rains today, how likely is it going to rain tomorrow?); or reasoning about the state of mind of the others (my opponent plays this card, is it because they have certain cards in their hand; hence they play this particular card as part of their strategy to defeat me? My wife does not answer my call, is she mad at me?). Therefore, the modelling, formalisation, and automation of such reasoning is an important subject for artificial intelligence.

Organisation

This dissertation will be organised as follows.

Chapter 2 We survey the literature that is relevant to or inspires our work.

Chapter 3 The notions from the field of game theory that we use throughout this dissertation are given in a formal and precise manner.

Chapter 4 The computational complexity of finding optimal pure strategies for different classes of extensive-form games is thoroughly studied.

Chapter 5 and 6 We turn our attention to games with incomplete information; the complexity and algorithms (and their optimisations) for these games are studied.

Chapter 7 We briefly touch on recursive reasoning and opponent modelling in games.

Contribution

Our main contributions are the following.

- We fill the gaps in the literature by proposing new polynomial-time algorithms and new hardness results, so that the full complexity landscape of finding robust strategies in games is now known and uniformly presented in one place for future reference.
- We introduce a new subclass of games with incomplete information, called combinatorial games with incomplete information, that minimally captures the essence of the notion of incomplete information. We fully justify its introduction and show that it is equally expressive as the generic model of extensive-form games of chance.
- By studying the optimisations of algorithms for games with incomplete information, we refine and develop techniques that can be applied to more general games (e.g. alpha-beta search under partial order).
- And near the end, we show how different kinds of knowledge about the opponents' reasoning and behaviour can be taken into account as opponent models, which can then be exploited by algorithms.

[2]For a ludic example, see "The One Where Everybody Finds Out", Episode 14, Season 5 of Friends.

Chapter 2

Related work

In this chapter, we provide a high-level overview of work relevant to this dissertation. More detailed and direct references will be given progressively along the natural narrative of the following chapters. Multiple references for the same subject are sorted by their potential (judged presumptuously by us) to promptly get readers familiar with the subject.

For well-established fields or well-known results in the literature, we prefer to cite books (especially textbooks) since they often provide a more holistic and pedagogical presentation of the subjects. We take great care to choose which books we cite so that readers new to a topic can take a leap of faith to dive into our recommendations.

2.1 Decision-making

This dissertation concerns decision-making, a field central to artificial intelligence. The standard reference on artificial intelligence is, no doubt, Russell and Norvig (2020), which comprehensively covers both classical AI (such as search, logic, and decision-making) and machine learning. As for the field of decision-making itself (which encompasses probabilistic reasoning, utility theory, game theory, automatic planning, etc.), the recent books in open access by Kochenderfer (2015) and Kochenderfer et al. (2022) seem to be good references; both provide copious pointers to the literature; the former focuses on theory and application, while the latter focuses on algorithmics, and presents highly readable Julia code for most important exact and approximate algorithms in the field of decision-making.

Here are some prominent models for decision-making under uncertainty.

Games Games from the field of game theory model multi-agent interaction. Non-cooperative [1] games come mainly in two categories: normal-form games, for one-round interaction during which agents pick their action concurrently; extensive-form games (EFGs), for multi-round interaction during which agents take turns to make decision. The reward at the end of a game is determined by the actions of each agent during the game; the goal of each agent is to maximise their own reward. Depending on whether an agent observes (or retains) the actions in the past, games can be with perfect or imperfect information. A game can also be with incomplete information, when players do not have common knowledge of

[1] "Non-cooperative" typically means agents do not necessarily have their interests aligned.

the game they play (e.g. a player does not know the reward function or available actions of the other players, such as in card games).

Markov decision processes MDPs (Puterman, 1994) are used to model sequential decision-making (typically with an infinite horizon) of a single agent in an environment described by a stochastic transition system. At each discrete time step, an agent takes an action, which brings them a certain amount of reward and causes the system to transition to another state according to some probability distribution. The goal of the agent is usually to maximise their total reward for a finite horizon or their discounted expected reward for an infinite horizon.

Contrary to games, MDPs (and their variants discussed below) do not explicitly store information about the history, i.e. the sequence of actions an agent has taken (and of observations they have received) in the past. In particular, the agent in an MDP can implement a history-dependent policy, i.e. take different actions in the same state of the system according to the history. On the other hand, the history in a game is explicitly represented by the position of the current node in the game tree. Hence, for modelling sequential decision-making, MDPs can be more compact than games, so are, in general, more difficult to solve (cf. Subsection 2.3.2).

Variants of MDP Like games, MDP also has many variants, depending on the number of agents and their observability on the state of the system: partially observable MDP (POMDP), which has one agent with partial observability (Kaelbling et al., 1998); multi-agent MDP, which has multiple cooperative[2] agents with full observability (Boutilier, 1996); decentralised POMDP (dec-POMDP), which has multiple cooperative agents with partial observability (Seuken and Zilberstein, 2008; Oliehoek and Amato, 2016); partially observable stochastic game (POSG), which has multiple non-cooperative agents with partial observability (Hansen et al., 2004). Reinforcement learning concerns situations in which an agent acts in an MDP with an unknown transition function (Sutton and Barto, 2018).

Propositional planning Propositional planning (Geffner and Bonet, 2013; Ghallab et al., 2016, 2004) is similar to MDP in spirit, but comes with a non-stochastic transition system compactly encoded by propositional variables. In addition, the typical goal of an agent is to reach a state instead of maximising accumulated reward. Similar to MDP, propositional planning can be classified into different categories, depending on whether an agent has full, partial, or no observability, and whether the transition system is deterministic or nondeterministic. The category particularly relevant to the topics of this dissertation is the one called contingent planning (agent with partial observability, nondeterministic environment) (Rintanen, 2004; Geffner and Bonet, 2013). It is worth mentioning that there is a variant of MDP called qualitative Dec-POMDP (Brafman et al., 2013) that extends contingent planning to the multi-agent setting.

Graph games Graph games (Apt and Grädel, 2011), also called games on graphs, are useful for synthesis and verification of reactive systems. In graph games, two non-cooperative agents with partial or full observability pick actions in a turn-based or concurrent manner on a graph with deterministic or stochastic transition; the

[2] "Cooperative" typically means agents share the same reward function; hence their interests are perfectly aligned.

goal of an agent is typically described by an ω-regular language over the vertices of the graph.

Kochenderfer et al. (2022) provide a particularly insightful vision about decision-making under uncertainty, which we reproduce here. Decision-making under uncertainty is a situation in which agents interact with an environment (and potentially each other) by choosing actions at discrete time steps to achieve a certain goal, based on their knowledge of the environment and past observations, under various sources of uncertainty.

Outcome uncertainty Agents are uncertain of the effects of their actions (e.g. systems with nondeterministic or probabilistic transitions).

Model uncertainty agents are uncertain of the transition rules of the environment (e.g. games with incomplete information; the problem setting of reinforcement learning).

State uncertainty Agents are uncertain of the true state of the environment (e.g. partial or no observability).

Interaction uncertainty Agents are uncertain of the behaviour of the other agents interacting in the same environment (e.g. EFGs; POSG; graph games).

2.2 Game theory

Among all models for decision-making under uncertainty, this dissertation concentrates on games, especially EFGs with incomplete information.

The rigorous treatment of game-theoretic subjects in this dissertation is highly inspired by Maschler et al. (2020), a modern and comprehensive textbook on game theory written in formal and rigorous style. It mainly covers non-cooperative game theory and cooperative game theory in depth from a mathematical perspective, with detailed proofs for nearly all results in the main text. For computational aspects, we can refer to Faliszewski et al. (2016), again an accessible introduction, which also covers (computational) social choice theory. Another viable choice is the book by Shoham and Leyton-Brown (2009), which presents game theory from the perspective of computer science, hence has more algorithmic and logical flavour and more emphases on game theory as a model for multi-agent systems. Finally, for examples of different kinds of games illustrating notions from non-cooperative game theory, one can turn to the textbook by Bonanno (2018).

For more advanced subjects on game theory, there are books that can serve as handy reference work, some more like an anthology than a textbook: Roughgarden and Iwama (2017) and Nisan et al. (2007) for algorithmic game theory; Brandt et al. (2016) for computational social choice; and Perea (2012, 2024) for epistemic game theory.

2.2.1 Solution concepts for games

The goal of studying a given game, be it in normal form or extensive form, is to predict the behaviour of the agents in a game, i.e. what strategies each agent will play if they participate in the game. A solution concept is a systematic way to output such a prediction for all games. During the long history of game theory (i.e. since the foundational work by von Neumann and Morgenstern (1994)), various solution concepts

have been proposed, some of which are presented below. Readers may also refer to Maschler et al. (2020, Chapter 7) and Shoham and Leyton-Brown (2009, Sections 3.3-4) for a quick overview.

Nash equilibrium Nash equilibrium (NE) is the central solution concept of game theory. An NE is a strategy profile (i.e. a combination of strategies for each player) such that no one has a strict unilateral incentive to deviate. Hence, NE is a solution concept about stability. Apart from stability, NE also has the desirable property of guaranteed existence: every finite game has an NE in mixed strategies (Nash, 1950, 1951).

Maxmin Maxmin, also called maximin, is another well-known solution concept. It predicts that every player aims to maximise the minimum reward they get against all possible strategies of their opponents, whence the name "maxmin". Hence, maxmin is a solution concept about safety and robustness since playing a maxmin strategy guarantees a certain amount of reward for a player, no matter what happens. Interestingly, in a zero-sum game, the notions of Nash equilibrium and maxmin coincide, in the sense that a mixed strategy profile is an NE if and only if each strategy in it is a mixed maxmin strategy.

Refinements of Nash equilibrium Nash equilibrium applied to EFGs may generate solutions with undesirable properties such as non-credible threat (i.e. not sequentially rational), or sensibility to opponent's small deviations from equilibrium. As a result, many refinements of NE have been proposed and studied in the literature, such as sequential equilibrium, perfect equilibrium, proper equilibrium, etc.; see the work by van Damme (1991) for a detailed exposition.

Rationality Very informally speaking, a player is rational if they choose the optimal action according to their belief about their opponents' possible behaviour, which does not necessarily have to be correct. If a player believes in their opponent's rationality, then they believe that their opponent never chooses actions that are never optimal according to their opponent's belief. As a result, this player will no longer choose actions that are only optimal against actions that will never be chosen by their opponent. Hence, belief in rationality leads to reduction of strategies; this reasoning can be repeated recursively, leading a fixed point considered to be the solution of a game. Assuming different types of belief or the order of recursion will lead to different solution concepts, such as rationalisability (Pearce, 1984), backward or forward induction (Perea, 2010), etc. These kinds of solution concepts are the subjects of epistemic game theory, which, rather than "solving games", prefers to study how players reason in games. See Perea (2012) for a more detailed treatment and comprehensive references of this field.

Behavioural game theory Human players do not have unlimited computational resources, nor are they completely rational due to cognitive limitations, emotions, and biases. As a result, the behaviour of humans in games is often not consistent with the predictions of usual solution concepts (Dhami, 2019, Chapter 1). The field of behavioural game theory takes these aspects into account and proposes solution concepts that are more adapted to strategic human choice, e.g. level-k models (Stahl and Wilson, 1995; Nagel, 1995) and cognitive hierarchy models (Camerer et al., 2004); see Dhami (2019, Chapter 2) for more models from this field.

For newcomers to game theory, it is important to keep in mind that there is no such thing as a *correct* or *wrong* solution concept: a solution concept is only a description or prediction, not a prescription. How well a solution concept can predict the outcome of a game depends fundamentally on how solid its underlying assumptions (e.g. level of rationality or computational resources) are with respect to the real players who play the game. In addition, a solution concept also expresses the desirable property (stability, robustness, rationality) we are looking for in the gameplay. In short, no single solution concept can be mindlessly applied to all types of games with the hope that it foresees perfectly how a game proceeds; one should always choose the right solution concept(s) to adapt to the classes of games and types of players in their problem setting. For more arguments about why we should care more about solution concepts besides NE, see Halpern (2011).

As for our work, we mainly cover the solution concept of maxmin, especially pure maxmin, for EFGs, since maxmin is well-suited for the game of Bridge.

2.2.2 Team games

In our work, we also study the complexity of maxmin in multi-agent settings, i.e. when MAX or MIN consists of multiple agents, who all have perfect recall but do not share observation or information. This setting, called *(two-)team games*, is worth a special mention since until quite recently, most of the literature on complexity or algorithms for equilibrium or maxmin focuses solely on games involving two agents.

Maxmin for team MAX is called *team maxmin equilibrium* (TME) in the literature, and was first proposed by von Stengel and Koller (1997). Almost twenty years later, Basilico et al. (2017); Celli and Gatti (2018) propose another maxmin notion called TMECor ("Cor" stands for "correlation"), which allows agents in team MAX to have access to a correlation device in order to coordinate in their mixed strategies (in a similar vein to correlated equilibrium; see Maschler et al. (2020, Chapter 8)). These works also study the inefficiency gap between NE, TME, and TMECor.[3] Seemingly, these works have stirred the community's interest in team games; many algorithms for computing TMECor of EFGs have been designed in recent years. To cite a few: Farina et al. (2018); Zhang et al. (2021); Zhang and Sandholm (2022); Farina et al. (2021); Zhang et al. (2023).

2.2.3 Games with public actions

One of our contributions is to propose a new subclass of games, which concern EFGs with incomplete information but with public actions and no chance node except at the root (i.e. the beginning of the game), where Nature draws a world (or types for players). We name this subclass *combinatorial game with incomplete information* (CGII) since it minimally generalises the notion of combinatorial games, which are two-player games of no chance and with perfect information (hence *a fortiori* public actions), to allow incomplete information.

The field of combinatorial game theory, which concerns the mathematical structure and analysis of these games, was established in the 70s by two books: *On Numbers and Games* (latest edition: Conway, 2000) and *Winning Ways* (latest editions: Berlekamp et al., 2001, 2003a,b, 2004). The latter multi-volume book still serves as a ludic and witty introduction to the field; see also Albert et al. (2019) for a modern and gentle

[3] Allowing more communication between agents of team MAX increases their maxmin value.

introduction and Siegel (2013) for an advanced treatment. For recent advances in the field, one may consult these collections of proceedings, appropriately named *Games of no Chance*: Nowakowski (1996, 2002); Albert and Nowakowski (2009); Nowakowski (2015); Larsson (2019).

Another motivation of ours for introducing CGII is to show that the difficulty (i.e. complexity) of solving two-player or two-team games does not come from hidden actions, but from incomplete information.

A third motivation (not originated from our own research, but from Kovarík et al. (2022)), is to highlight the distinction between public and private actions. According to Kovarík et al. (2022), this distinction, essential for recent search algorithms, is partially lost when we model sequential multi-agent interaction with EFGs, which do not explicitly tell whether an action is public or not (but implicitly by using information sets of players). In their work, they propose an alternative model for stochastic games that makes this distinction prominent, and show how to transform such models to augmented EFGs and vice versa.

2.3 Computational complexity

The standard reference on computational complexity is Arora and Barak (2009), which contains the definitions of all notions from the theory of computation that we use in this dissertation (the notion of reduction, complexity classes such as NP and PSPACE, canonical complete problems for these classes, etc.), and much more. It can be complemented by the book (in open access but written in French) by Perifel (2014), which painstakingly fills the gaps in Arora and Barak (2009) by giving formal definitions and detailed proofs. Readers new to the theory of computation can also turn to Sipser (2012) and Moore and Mertens (2011). Both are very accessible introductions: the former one is formal and gets to the point quickly; the latter one is more casual and does not hesitate to discuss the connections between the theory of computation and other subjects, especially natural sciences. After these two books, one may turn to Kozen (2006) for more advanced subjects, clearly presented with lucid proofs.

2.3.1 Complexity of games

Most work in the literature on the computational complexity of games concerns the complexity of finding Nash equilibria, especially for normal-form games (e.g. Gilboa and Zemel (1989); Daskalakis et al. (2009), who show the PPAD-completeness of computing NE). For more references, one may consult the introduction by Conitzer and Sandholm (2008), who also show that it is NP-complete to decide whether NEs with certain natural properties exist.

Koller and Megiddo (1992); Koller et al. (1996); von Stengel (1996) made the first major steps towards understanding the complexity landscape of solving two-player zero-sum EFGs. Apart from studying the complexity of computing pure, behaviour, and mixed maxmin strategies for some classes of EFGs, they also give polynomial-time algorithms for computing behaviour maxmin strategies of EFGs with perfect recall, based on linear programming. McMahan et al. (2003) propose a double-oracle algorithm for computing optimal strategies for Markov decision processes with adversarial cost functions, which can also be regarded as a polynomial-time algorithm for computing the mixed maxmin strategies of a normal-form game. Bosanský et al. (2014)

combine these ideas to propose an algorithm for zero-sum EFGs with perfect recall, which is efficient when optimal mixed strategies have small supports.

Building on the work by Koller and Megiddo (1992), Gimbert et al. (2020) and Zhang et al. (2023) study the complexity of TME and TMECor, thereby yielding a relatively complete picture of the complexity of behaviour and mixed maxmin for two-team EFGs.

Peterson et al. (2001) study the complexity of compactly represented EFGs. However, they define games as alternating Turing machines with different kinds of specifications (number of alternations, amount of space, private or public band, etc.). They also invent a formalism called dependency quantified Boolean formulae, generalising the well-known formalism of quantified Boolean formulae (which compactly represents two-player EFGs of no chance and with perfect information) to allow multiple agents of MAX, each having their private information. The complexity of deciding the truth value of such a formula is proved to be NEXP-complete, in contrast with the PSPACE-completeness for QBF.

Finally, Bonnet et al. (2013) show that perfect information trick-taking card games, which include double dummy Bridge (i.e. Bridge played with no hidden hand) and many other card games, are PSPACE-complete with respect to the number of players, suits, and ranks encoded in unary. However, the exact complexity of Bridge itself, a team-vs-player game, is still unknown. For complexity results of puzzle games and board games, and a unifying framework to analyse such results, see the monograph by Hearn and Demaine (2009).

2.3.2 Complexity of other models for decision-making

Parallelly, the complexity of graph games for different observability and objectives has been thoroughly studied: see the surveys by Chatterjee and Henzinger (2012) and Chatterjee et al. (2013).

The complexity (and computability) of automatic planning is also well-established. One may refer to Mundhenk et al. (2000) for the complexity of finite horizon MDP and POMDP (with different observability, stationary or time/history-dependent policies, short or long horizon); Madani et al. (2003) for the undecidability of infinite horizon POMDP; Bernstein et al. (2002) for the complexity of Dec-POMDP; Brafman et al. (2013) for the complexity of qualitative Dec-POMDP; Goldsmith and Mundhenk (2007) for the complexity of POSG; and Rintanen (2004) for the complexity of propositional planning with full/no/partial observability and deterministic/nondeterministic transitions.

2.4 Search algorithms for games

Algorithms for playing games have been studied since the birth of the field of game theory and artificial intelligence; see Russell and Norvig (2020, Chapter 5) for an overview. Notable examples include the minimax algorithm (Zermelo, 1913), alpha-beta search (Knuth and Moore, 1975; Pearl, 1982; Marsland, 1986), the SSS* algorithm (Stockman, 1979), iterative deepening (Korf, 1985), proof-number search (Allis et al., 1994), Monte Carlo tree search (Coulom, 2006; Kocsis and Szepesvári, 2006; Browne et al., 2012; Swiechowski et al., 2023), and (deep) reinforcement learning (Sutton and Barto, 2018).

These techniques have been very successful in building AIs for games with perfect information, e.g. Chess (Campbell et al., 2002; Haworth and Hernandez, 2021), Checkers (Schaeffer et al., 2007; Schaeffer, 2008), Go (Silver et al., 2016, 2017), and even Atari games (Schrittwieser et al., 2020).

2.4.1 Games with incomplete information

A good introduction to games with incomplete information is provided, as you can guess, by Maschler et al. (2020, Chapter 9); one may also consult Bonanno (2018, Part V).

Compared to games with perfect information, games with imperfect (and in particular, incomplete) information have received relatively little attention. However, there is now a growing interest in this area since it is a natural continuation from games with perfect information; we may cite a few examples of recent research attacking such games: Bridge (Ginsberg, 2001; Cazenave and Ventos, 2020), Skat (Kupferschmid and Helmert, 2006; Buro et al., 2009; Rebstock et al., 2019; Edelkamp, 2020), Hearts and Spades (Sturtevant and White, 2006), Stratego (Pérolat et al., 2022), and most notably Poker (Bowling et al., 2017; Brown and Sandholm, 2017, 2019).

Important algorithms in this field include perfect information Monte Carlo search (Levy, 1989; Long et al., 2010), Information Set MCTS (Cowling et al., 2012), and counterfactual regret minimisation (Zinkevich et al., 2007).

2.4.2 Games with public actions

Of particular interest to us is the problem of finding pure optimal strategies for games with incomplete information and public actions, with Bridge as our main inspiration and target (first advocated by Levy (1989)). Along this line of research, Frank and Basin (1998) propose the best-defence model, which consists in approximating a game with incomplete information by a game with one-sided incomplete information in which MIN has complete information. They put forward the notions of strategy fusion and non-locality to explain why the traditional minimax algorithm (i.e. backward induction) does not solve such games correctly. Subsequently, Frank and Basin (2001) showed that finding optimal pure strategies for these games is NP-complete, even when all actions are public. Later, Ginsberg (2001) proposes the first exact algorithm for this problem without proving its correctness. Many works, including ours, build on their algorithm and apply it to different games; for instance, Cazenave and Ventos (2020) and Cazenave et al. (2021) for Bridge.

Bridge

On the matter of computer Bridge, besides the references mentioned before, a summary of recent advances can be found in the PhD thesis by Bethe (2021, Section 2.3) or the article by Khemani et al. (2018, Section 3). Recently, NukkAI, the sponsor behind this PhD thesis, organised a challenge in which their Bridge robot (based on the algorithms developed by Cazenave and Ventos (2020) and Cazenave et al. (2021)) won against 8 human Bridge champions in a restricted setting; see Rousset (2022, in French only) for a technical overview of this achievement.

Multi-objective games

A model that has very similar spirit and structure to games with one-sided incomplete information and public actions is the one of multi-objective games. To compute robust strategies for such games, Dasgupta et al. (1996) propose multi-objective game tree search with alpha-beta-pruning-like optimisations. Their work is part of the book by Dasgupta et al. (1999), which covers many other topics on multi-criteria optimisations and decision-making, such as multi-objective path planning. A more recent monograph on multi-objective decision-making is given by Roijers and Whiteson (2017).

2.5 Opponent models

Part of our work concerns the complexity of, and algorithms for, computing optimal strategies when it is assumed that the opponents' strategies are taken from a known, restricted set. Such known strategies are referred to in the literature as opponent models. Behavioural game theory concerns in particular models of human behaviour in games; see Subsection 2.2.1.

Opponent models can come in diverse forms. Iida et al. (1993, 1994) propose opponent models for games with perfect information, where models are given by the evaluation function and the search depth of the opponent. Sturtevant et al. (2006) propose opponent models given by opponent's preferences over the outcomes of a game. Rebstock et al. (2019) use opponent models learnt from human games for imperfect information (card) games. A survey of opponent modelling approaches is also provided by Albrecht and Stone (2018). Our work is related to these in the sense that we assume opponent models to be given (called "type-based reasoning" by Albrecht and Stone (2018, Section 4.2)). However, an important stream of work also studies the *learning* of opponent models; we refer the reader to the survey by Nashed and Zilberstein (2022).

An important class of opponent models is that of *recursive* models, where MAX searches a strategy (at level k) assuming that MIN themselves searches a strategy (at level $k-1$) assuming that MAX searches. . . , etc., down to level 0. Such models have been studied in behavioural game theory (Dhami, 2019) to capture human reasoning. For instance, Camerer et al. (2004) propose a cognitive hierarchy model in which an opponent's level is modelled by a Poisson distribution over levels $k-1, \ldots, 0$; they also validate this model against empirical data. Wright and Leyton-Brown (2019) assess the relevance of various modelling assumptions for level 0. De Weerd et al. (2013) assess the efficiency of reasoning with recursive models by simulation. Such recursive models are also used in epistemic game theory to define notions such as common belief in rationality (Perea, 2012).

Recursive models are also considered for cooperative planning. In particular, interactive POMDPs are a framework for collaborative decision-making in partially observed environments (Gmytrasiewicz and Doshi, 2005; Doshi et al., 2020). In this model, a level-k agent optimises their behaviour given a distribution over (partially observed) physical states and over other agents' models at level $k-1$. Interestingly, optimal behaviours at level k can be computed iteratively by solving a sequence of POMDPs, where at each iteration the other agents' model can be considered as part of the environment. For human-agent collaboration, You et al. (2023) propose a recursive model with bounded depth, whereby an agent plans (at level 2) a best response to a mixture of possible strategies for a human, with these human strategies themselves defined (at level 1) by planning assuming that the agent is controlled by the human (at level 0).

Chapter 3

Background on game theory

In this chapter, we introduce notions from game theory that we will use throughout this dissertation. The goal of this chapter is not to be an introduction to game theory, but to fix the terminology and notation of the notions we will use. For (much) more details on game theory, one can refer to the textbook by Maschler et al. (2020), on which most of this chapter is based.

Furthermore, note that the definitions of relevant notions from graphs and sets are given in the appendix (Section A.1 and Section A.3).

3.1 Games and strategies

In this section, we define the central subject of our work: extensive-form games. We also define strategies and the solution concept of maxmin.

3.1.1 Extensive-form games

Games with perfect information

For pedagogical reason, we begin with the definition of extensive-form games (abbreviated as EFGs) with perfect information, which are special cases of EFGs with imperfect information that will be introduced right after.

Definition 3.1.1 (EFG with perfect information)**.** *An* extensive-form game with perfect information *is a tuple* $G = (T, P, (V_i)_{i \in P}, u)$*, where:*

- $T = (V, E, r)$ *is a finite tree, called the* game tree*;*
- P *is a finite set, called the set of* players*;*
- $(V_i)_{i \in P}$ *is a partition of the set of internal vertices* $V \setminus \mathrm{L}(T)$*;*[1]
- $u : \mathrm{L}(T) \to \mathbb{R}^{|P|}$ *is the* utility function*, which assigns to each leaf one real number for each player in* P*, called the* utility *(or* reward*, or* payoff*) for the player at the leaf.*

[1] We use $\mathrm{L}(T)$ to denote the set of all leaves of the tree T; see Section A.1.

Throughout this dissertation, we use natural numbers to denote the players: $P = \{1, 2, \ldots, |P|\}$. When there are only two players, we also write $P = \{+, -\}$ and call them *player MAX* and *player MIN*, respectively. For $i \in P$, we also write u_i for the i-th component of u. Thus, u_i is the utility function of player i.

Definition 3.1.2 (Decision maker)**.** *Let $v \in V \setminus \mathrm{L}(T)$. If $v \in V_i$ for a certain $i \in P$, we say that v is a* decision node *of player i, and i is said to be the* decision maker *of v.*

Remark. *Since $(V_i)_{i \in P}$ is a partition of all internal vertices, each internal vertex v has a unique decision maker.*

EFGs are said to be in extensive-form because they model sequential interaction among the players by a tree. In this dissertation, we are only interested in such games, the game tree of which is either defined explicitly by the input or implicitly (and compactly) by the game rules.

Informally, an EFG with perfect information is played in the following way. The game begins at the root of the tree. Then, at each turn, the decision maker of the current node chooses a child of that node, which will be the current node for the next turn. The game proceeds in this way until a leaf is reached. Then each player receives a reward determined by applying their utility function to this leaf. Such a game is said to be with perfect information because every player, when it is their turn to make a decision, knows perfectly the current node (and in particular its position in the game tree).

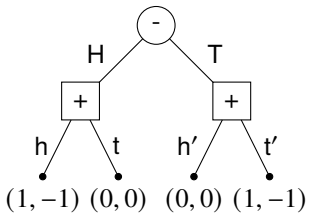


Figure 3.1: Matching Pennies with perfect information.

Example. *An example of EFG with perfect information is given in Figure 3.1. This game has two players: MAX (+) and MIN (−). The decision maker of each internal node is indicated by both a symbol representing the player and their form.*[2] *The game is called* matching pennies *since the goal of MAX is to match the side of a penny chosen by MIN. For example, if MIN picks heads and so does MAX, then MAX receives* 1 *as reward while MIN receives* −1*. On the other hand, if they choose different faces, then both receive* 0*.*

The semantics of this game is that MAX picks between heads and tails after *observing MIN's choice. In particular, MAX knows where they are in the game tree when it is their turn to make a decision. Hence, MAX can use the strategy "pick the child connected by the edge labelled by* h *when MIN picks* H*, and pick* t′ *when MIN picks* T*". to receive a guaranteed payoff of* 1*, regardless of what MIN plays.*

Remark. *Notice that in the previous example, the reward for the players always sums up to* 0 *at every leaf. A game with this feature is said to be* zero-sum*.*

[2]Throughout this dissertation, nodes owned by MAX and MIN will be drawn as squares and circles/ellipses, respectively, in accordance with the standard practice in the literature.

Informally, a strategy of a player in an EFG with perfect information is a mapping that assigns to each decision node of the player a child of that node; we postpone the formal definition.

Chance moves can be incorporated into the current definition by introducing one special player called *Nature*, who chooses its moves at each of its decision node with a probability distribution that is common knowledge to all players. Again, we will see a more formal definition later.

Games with imperfect information

For many decision-making problems that we desire to model as games, decision-makers do not always have perfect information. To model this aspect, we need to introduce the concept of information set, which is a way to express the idea that a player cannot distinguish one situation from another.

Definition 3.1.3 (Information set and available actions)**.** *Let $G = (T, P, (V_i)_{i\in P}, u)$ be a game in extensive form. An* information set *of player $i \in P$ is a pair $\langle IS_i, A_i\rangle$, where*

- *$IS_i \subseteq V_i$ is a set of decision nodes of player i such that all vertices in IS_i have the same number of children, denoted by c;*
- *$A_i = \{a_1, \ldots, a_c\}$, called the set of* available actions *of player i at IS_i, is a partition of the $c|IS_i|$ children of the vertices in IS_i, such that for $1 \le j \le c$, action a_j contains exactly one child of each of the vertices in IS_i.*

In other words, each vertex from the same information set $\langle IS_i, A_i\rangle$ has c children:

$$\forall v \in IS_i, |\mathrm{C}(v)| = c;$$

and each of the c children of each vertex is labelled by a distinct action a from A_i:

$$\forall a \in A_i, \forall v \in IS_i, |a \cap \mathrm{C}(v)| = 1.$$

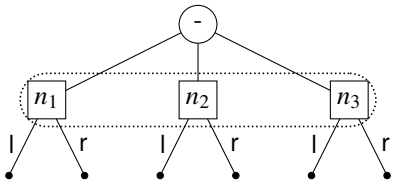


Figure 3.2: An example to illustrate the idea of information set. The rewards are omitted for simplicity.

Example. *In the game in Figure 3.2, the three nodes of MAX are in the same information set, indicated by a bubble drawn by a dotted line. Formally, MAX has an information set $\langle IS_+, A_+\rangle$, where $IS_+ = \{n_1, n_2, n_3\}$, (hence $c = 2$, meaning that each node in IS_+ has exactly 2 children), and $A_+ = \{\mathsf{l}, \mathsf{r}\}$, where l is the set of the left child of the nodes in IS_+, while r is the set of their right child (in particular, $|\mathsf{l}| = |\mathsf{r}| = |IS_+| = 3$).*

Vertices from the same information set of a player are regarded as indistinguishable by that player: when they are asked to make a decision, they are only informed that this information set is reached, without knowing exactly which vertex is reached. We will

see later in the definition of strategies that a player can only choose one action at their information set, which dictates the next node to be reached, depending on which vertex in the information set the current node is. This allows us to impose the constraint that a player cannot base their decision on the information they do not possess.

Armed with the notion of information set, we can define a more general class of EFG.

Definition 3.1.4 (EFG with imperfect information)**.** *An* EFG with imperfect information *is a tuple*

$$G = \left(T, P, (V_i)_{i\in P}, u, \left(\langle IS_i^j, A_i^j\rangle\right)_{i\in P}^{j=1,\dots,k_i}\right),$$

where:

- $T = (V, E, r)$ *is a finite game tree;*
- P *is a set of players;*
- $(V_i)_{i\in P}$ *is a partition of* $V \setminus \mathrm{L}(T)$*;*
- $u : \mathrm{L}(T) \to \mathbb{R}^{|P|}$ *is a utility function;*
- *for each player* $i \in P$, $\left(\langle IS_i^j, A_i^j\rangle\right)^{j=1,\dots,k_i}$ *is a set of information sets of player* i *such that* $(IS_i^j)^{j=1,\dots,k_i}$ *forms a partition of* V_i.

In the following, we will abuse the language and refer to a set IS_i^j as an information set of player i, leaving implicit the set A_i^j of actions available at IS_i^j. We denote the set of all information sets of player i in an EFG with imperfect information by

$$\mathcal{IS}_i \coloneqq \{IS_i^1, \dots, IS_i^{k_i}\}.$$

We also emphasise that, by definition, the union of the sets in $\mathcal{IS}_i$ yields V_i, the set of all decision nodes of player i. Notice that the main difference between an EFG with perfect information and one with imperfect information is that in the latter, each player's decision nodes are partitioned into their information sets.

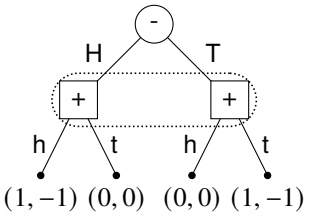


Figure 3.3: Matching Pennies.

Example. *The EFG with imperfect information in Figure 3.3 is very similar to the EFG with perfect information in Figure 3.1; the only difference is that in Figure 3.3, MAX's decision nodes are now in the same information set. Hence, the semantics of this game is that MAX now chooses between heads and tails* without *observing MIN's choice. In that particular case, one can also imagine that MAX and MIN* concurrently *or* simultaneously *pick their action. Notice that MAX can no longer guarantee a reward of* 1*, since their information set imposes the constraint that they must choose either* h *at both their decision nodes, or* t *at both.*

Definition 3.1.5 (Perfect information)**.** *In an EFG with imperfect information, a player $i \in P$ is said to have* perfect information *(PI) if every information set of i is a singleton.*

In particular, an EFG with perfect information can be regarded as an EFG with imperfect information in which all players have perfect information. Hence, in the following, when we say EFG, we always mean EFG with imperfect information; when we actually refer to an EFG with perfect information, we will say so explicitly.

3.1.2 Strategies and outcomes

We now define the notion of strategies and outcomes. Let

$$G = \left(T, P, (V_i)_{i\in P}, u, \left(\langle IS_i^j, A_i^j\rangle\right)_{i\in P}^{j=1,\dots,k_i}\right)$$

be an arbitrary EFG with imperfect information in this and the following subsections.

Pure strategies

Definition 3.1.6 (Pure strategy)**.** *Let $i \in P$ be a player in a game G. A* pure strategy *of player i is a mapping*

$$s_i : \mathcal{IS}_i \to \bigcup_{1\le j\le k_i} A_i^j$$

such that for every information set $IS_i^j \in \mathcal{IS}_i$, $s_i(IS_i^j) \in A_i^j$. The set of all pure strategies of player i in the game G will be denoted by Σ_i^{P}.

In other words, a pure strategy of a player i maps each information set of player i to one of their available actions at that information set. We say that player i *plays* or *implements* a pure strategy $s_i \in \Sigma_i^{\mathrm{P}}$ in the game if at every information set of player i, they pick the action prescribed by s_i at that information set.

Notice that since we only consider games with a finite game tree, the number of pure strategies of a player is finite, although it can be extremely large (exponential in the size of the game tree).

Definition 3.1.7 (Pure strategy profile)**.** *A* pure (strategy) profile *in a game G is a vector containing one pure strategy for each player:*

$$s = (s_1, s_2, \dots, s_{|P|}) \in \prod_{i=1}^{|P|} \Sigma_i^{\mathrm{P}},$$

where $\prod$ denotes the Cartesian product of sets. The set of all pure strategy profiles will be denoted by Σ^{P}.

Every pure strategy profile uniquely determines a traversal of the game tree and the leaf to be reached. Let $s = (s_1, s_2, \dots, s_{|P|}) \in \Sigma^{\mathrm{P}}$ be a pure strategy profile. This strategy profile determines a unique path from the root r to a leaf

$$v_0 = r, v_1, v_2, \dots, v_{n-1}, v_n \in \mathrm{L}(T)$$

such that for all $k < n$, v_{k+1} is the unique child of v_k that is in the action chosen by the decision maker $i \in P$ of v_k at their information set containing v_k according to their pure strategy s_i.[3] This well-defined path is called the *playout* under the profile s.

[3] More formally, let $IS_i^j \in \mathcal{IS}_i$ be the unique information set of i that contains v_k and $a = s_i(IS_i^j) \in A_i^j$ be the action chosen by player i at IS_i^j under pure strategy s_i. Then v_{k+1} is the unique vertex in $a \cap \mathrm{C}(v_k)$, which is a singleton set by the definition of an available action at an information set.

Definition 3.1.8 (Outcome)**.** *Let $s \in \Sigma^{\mathrm{P}}$ be a pure strategy profile in a game G and*

$$v_0 = r, v_1, v_2, \ldots, v_{n-1}, v_n \in \mathrm{L}(T)$$

be the unique path from the root to a leaf determined by the profile in the aforementioned way. Then the leaf v_n is called the outcome *of the game G under the strategy profile s. For all $i \in P$, $u_i(v_n)$ is called the* utility/reward/payoff *for player i under s.*

Hence, a utility function u uniquely defines a mapping from Σ^{P} to $\mathbb{R}^{|P|}$, which we will abusively denote by the same notation u. Hence, $u(s) \coloneqq u(l)$, where $l \in \mathrm{L}(T)$ is the outcome under $s \in \Sigma^{\mathrm{P}}$.

Mixed strategies

Definition 3.1.9 (Mixed strategy)**.** *Let $i \in P$ be a player in a game G. A* mixed strategy *of player i is a probability distribution over Σ_i^{P}, i.e. a mapping $\sigma_i : \Sigma_i^{\mathrm{P}} \to [0, 1]$ such that*

$$\sum_{s_i \in \Sigma_i^{\mathrm{P}}} \sigma_i(s_i) = 1.$$

The set of all mixed strategies of player i will be denoted by Σ_i^{M}.

We say that player *i plays* or *implements* a mixed strategy $\sigma_i \in \Sigma_i^{\mathrm{M}}$ in the game, if they choose a pure strategy $s_i \in \Sigma_i^{\mathrm{P}}$ according to the probability distribution σ_i, then play s_i in the game. Notice that pure strategies can be regarded as special cases of mixed strategies (by which a player chooses a certain pure strategy with probability 1).

Definition 3.1.10 (Mixed strategy profile)**.** *A* mixed (strategy) profile *in a game G is a vector containing one mixed strategy for each player:*

$$\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{|P|}) \in \prod_{i=1}^{|P|} \Sigma_i^{\mathrm{M}}.$$

The set of all mixed strategy profiles will be denoted by Σ^{M}.

By definition, the set of all mixed strategies is the simplex of the set of all pure strategies: $\Sigma_i^{\mathrm{M}} = \Delta(\Sigma_i^{\mathrm{P}})$. For profiles, we can also view Σ^{M} as a subset of $\Delta(\Sigma^{\mathrm{P}})$ since a mixed strategy profile $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{|P|}) \in \Sigma^{\mathrm{M}}$ induces a probability distribution over the set of pure strategy profiles Σ^{P} in the following way: for all pure strategy profiles $s = (s_1, s_2, \ldots, s_{|P|}) \in \Sigma^{\mathrm{P}}$, we can abuse the notation and write

$$\sigma(s) \coloneqq \prod_{i \in P} \sigma_i(s_i).$$

Then $\sum_{s \in \Sigma^{\mathrm{P}}} \sigma(s) = 1$, hence σ is a probability distribution over Σ^{P}. In particular, this means σ induces a probability distribution over the outcomes of the game (i.e. the reachable leaves).

Remark. *Note that Σ^{M} is a* strict *subset of $\Delta(\Sigma^{\mathrm{P}})$. For example, consider $\sigma = \frac{1}{2}(s_+, s_-) + \frac{1}{2}(s'_+, s'_-)$ with $s_i, s'_i \in \Sigma_i^{\mathrm{P}}$ for $i \in \{+, -\}$, which intuitively means for half of the time MAX plays s_+ while MIN plays s_-, and another half of the time MAX plays s'_+ while MIN plays s'_-. Then σ is an element of $\Delta(\Sigma^{\mathrm{P}})$ but not an element of Σ^{M}, since under no mixed strategy of MAX and MIN can they achieve the* perfect coordination *in σ. The difference between Σ^{M} and $\Delta(\Sigma^{\mathrm{P}})$ is essentially the difference between the solution concepts TME and TMECor; see Celli and Gatti (2018) and other references about team games in Subsection 2.2.2.*

Viewing Σ^{M} as a subset of $\Delta(\Sigma^{\mathrm{P}})$, we define the notion of expected utility.

Definition 3.1.11 (Expected utility)**.** *Let $\sigma \in \Sigma^{\mathrm{M}} \subsetneq \Delta(\Sigma^{\mathrm{P}})$ be a mixed strategy profile. Then the* expected utility/reward/payoff *for player i under σ, denoted by $\mathcal{U}_i(\sigma)$, is defined to be*

$$\mathcal{U}_i(\sigma) \coloneqq \sum_{s \in \Sigma^{\mathrm{P}}} \sigma(s) u_i(s).$$

In other words, the expected utility for a player under a mixed profile is the expectation of their utility if every player randomly chooses a pure strategy (to play throughout the game) according to the probability distribution represented by their mixed strategy in the profile.

Behaviour strategies

Instead of mixing pure strategies of a player, we may also consider mixing their choice of actions at each of their information sets. This yields a new notion of strategy.

Definition 3.1.12 (Behaviour strategy)**.** *Let $i \in P$ be a player in a game G. A* behaviour strategy *of player i is a mapping*

$$\pi_i : \mathcal{IS}_i \to \bigcup_{1 \le j \le k_i} \Delta(A_i^j)$$

such that for every information set $IS_i^j \in \mathcal{IS}_i$, $\pi_i(IS_i^j) \in \Delta(A_i^j)$. The set of all behaviour strategies of player i in the game G will be denoted by Σ_i^{B}.

We say that player *i plays* or *implements* a behaviour strategy $\pi_i \in \Sigma_i^{\mathrm{B}}$ in the game, if at every information set of player i, they pick their actions randomly according to the probability distribution prescribed by π_i at that information set. Notice that pure strategies can be regarded as special cases of behaviour strategies (by which a player chooses, at every information set, a certain action available at this information set with probability 1).

Definition 3.1.13 (Behaviour strategy profile)**.** *A* behaviour (strategy) profile *in a game G is a vector containing one behaviour strategy for each player:*

$$\pi = (\pi_1, \pi_2, \ldots, \pi_{|P|}) \in \prod_{i=1}^{|P|} \Sigma_i^{\mathrm{B}}.$$

The set of all behaviour strategy profiles will be denoted by Σ^{B}.

Like a mixed strategy profile, a behaviour strategy profile also induces a probability distribution over the outcomes of the game; hence we can define the expected utility/reward/payoff for a player under a behaviour strategy profile as the expectation of the utility/reward/payoff for the player under this induced probability distribution over the outcomes. We will not detail the construction here, see for example Maschler et al. (2020, Chapter 6).

Recapitulation of notation

We will consistently use the following notation throughout this dissertation.

- The set of pure/mixed/behaviour strategy profiles is denoted by Σ^{P}, Σ^{M}, Σ^{B}, respectively. An arbitrary pure/mixed/behaviour strategy profile is denoted by s, σ, π, respectively.
- When the type of strategies (pure/mixed/behaviour) under consideration is unspecified or inessential, we suppress the superscript and simply write Σ for the set of profiles and ς for an arbitrary profile.
- For a set of profiles (with or without superscript), a subscript $i \in P$ indicates the set of strategies of player i (e.g. Σ_i^{P}), and $-i$ indicates the set of profiles for all the other players[4] (e.g. Σ_{-i}^{P}). For example, we have

$$\Sigma = \prod_{j=1}^{|P|} \Sigma_j \cong \Sigma_i \times \Sigma_{-i},$$

  which holds for all $i \in P$ and all superscripts.
- For an arbitrary (pure/mixed/behaviour) strategy profile, a subscript $i \in P$ indicates the strategy of player i in this profile (e.g. π_i), and $-i$ indicates the strategy profile for all the other players in this profile (e.g. π_{-i}).
- A utility function $u : \mathrm{L}(T) \to \mathbb{R}^{|P|}$ uniquely defines an expected utility function on the set of pure/mixed/behaviour strategy profiles, which we denote by $\mathcal{U}$ in all cases.[5]

3.1.3 Games of chance

Real-life games can contain various sources of randomness and chance factors, such as dice rolls, coin flips, cards shuffles, etc. To model such random events, we may introduce a special player called *Nature*, who owns some internal nodes of a game tree, called chance nodes. The idea is that at each chance node, the successor is chosen (by Nature) according to some probability distribution that is common knowledge among the players (i.e. everybody knows the distribution, and everybody knows everybody else knows the distribution, ad infinitum). In addition, the drawing of successors at each chance node should be independent.

To this end, Nature, a player that we will represent by the index 0, should have perfect information in the game and implement a behaviour strategy that is common knowledge to all other players. Hence, the following definition.

Definition 3.1.14 (EFG of chance and of no chance)**.** *An* EFG of chance *(and with imperfect information) is a tuple*

$$G = \left(T, P, (V_i)_{i\in\{0\}\cup P}, (p_v)_{v\in V_0}, u, \left(\langle IS_i^j, A_i^j\rangle\right)_{i\in P}^{j=1,\dots,k_i}\right),$$

where:

[4] They are also referred to as the *opponents* or *adversaries* of player i.

[5] Since pure strategy profiles can also be interpreted as mixed strategy profiles, $\mathcal{U}$ is well-defined on pure strategy profiles.

- $T = (V, E, r)$ *is a finite game tree;*
- P *is a set of (non-Nature) players;*
- $(V_i)_{i \in \{0\} \cup P}$ *is a partition of* $V \setminus \mathrm{L}(T)$*;*
- *for every* v *from the set* V_0 *(called the set of* chance nodes*),* p_v *is a probability distribution over the children of* v*:* $p_v \in \Delta(\mathrm{C}(v))$*;*
- $u : \mathrm{L}(T) \to \mathbb{R}^{|P|}$ *is a utility function;*
- *for each player* $i \in P$*,* $\left(\langle IS_i^j, A_i^j \rangle\right)^{j=1,\dots,k_i}$ *is a set of information sets of player* i *such that* $(IS_i^j)^{j=1,\dots,k_i}$ *forms a partition of* V_i*.*

If a game has no chance node (i.e. $V_0 = \emptyset$*), it is called an* EFG of no chance*.*

Hence, by definition, games of no chance form a strict subclass of games of chance.

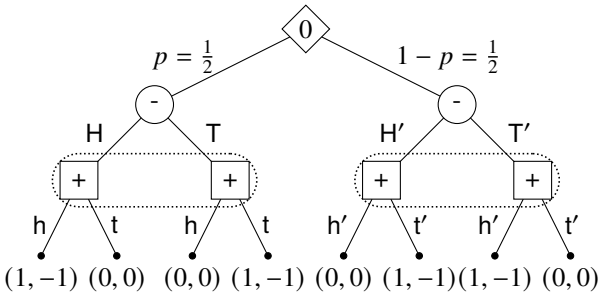


Figure 3.4: Random Matching Pennies.

Example. *The EFG of chance in Figure 3.4 has only one chance node, the root, at which Nature chooses uniformly at random between the left and right branches. If the left branch is chosen, then MAX and MIN play the game Matching Pennies (cf. Figure 3.3); otherwise, they play the version of Matching Pennies with reward inverted (hence MAX should not match MIN's choice).*

Remark. *Notice that in the previous example, the choice of Nature is observable by both players, since no player has an information set that contains vertices in both the subgames. In general, this needs not be the case; the choice of Nature at a chance node can be revealed only to some players or to none at all.*

EFG of chance (with imperfect information) is the most general model of games in extensive form that we will consider. In the following, we regard both EFGs of no chance and EFGs with perfect information as degenerate cases of EFGs of chance. In addition, we simply say *game* for EFG of chance, when there is no danger of confusion.

Moreover, Definition 3.1.14 only serves as a reference; in the following, we will not describe a game in this cumbersome notation, but rather by giving a figure of it (e.g. Figure 3.4) or by describing the gameplay.

3.1.4 The solution concept of maxmin

Definition 3.1.15 (Maxmin value)**.** *The* maxmin value *for player i in a game G with respect to a set of strategies Σ_i of player i and a set of profiles Σ_{-i} of their opponents is defined to be*

$$\underline{v}_i(\Sigma_i, \Sigma_{-i}) := \max_{\varsigma_i \in \Sigma_i} \min_{\varsigma_{-i} \in \Sigma_{-i}} \mathcal{U}_i(\varsigma_i, \varsigma_{-i}).$$

Maxmin is a security/robustness concept: $\underline{v}_i(\Sigma_i, \Sigma_{-i})$ tells us the largest payoff that player i can guarantee by playing a strategy in Σ_i, when their opponents' strategy profile ranges over Σ_{-i}.

Definition 3.1.16 (Best response and maxmin strategy)**.** *Let $\varsigma_i^* \in \Sigma_i$ and $\varsigma_{-i}^* \in \Sigma_{-i}$.*

- *ς_{-i}^* is called a* best response *to ς_i^* in Σ_{-i}, if*

$$\mathcal{U}_i(\varsigma_i^*, \varsigma_{-i}^*) = \min_{\varsigma_{-i} \in \Sigma_{-i}} \mathcal{U}_i(\varsigma_i^*, \varsigma_{-i}).$$

- *ς_i^* is called an* optimal strategy *or a* maxmin strategy *in Σ_i against Σ_{-i}, if ς_i^* achieves the maxmin value $\underline{v}_i(\Sigma_i, \Sigma_{-i})$, i.e.*

$$\min_{\varsigma_{-i} \in \Sigma_{-i}} \mathcal{U}_i(\varsigma_i^*, \varsigma_{-i}) = \underline{v}_i(\Sigma_i, \Sigma_{-i}).$$

The maxmin value for player i depends on the set of strategies we allow them to implement. In addition, this value is increasingly monotone on its first argument Σ_i (with respect to set inclusion). Usual choices for Σ_i are given by $\Sigma_i^{\mathrm{P}}, \Sigma_i^{\mathrm{M}}, \Sigma_i^{\mathrm{B}}$, for which the maxmin is named pure/mixed/behaviour maxmin. In our dissertation, we mostly focus on pure maxmin.

Similarly, the maxmin value depends and is decreasingly monotone on its second argument Σ_{-i}, the set of strategy profiles allowed for the opponents of player i. The largest such set we will consider is Σ_{-i}^{M}, the set of profiles formed by all combinations of the other players' mixed strategies.

Zero-sum games

Games with only two players (not counting Nature, i.e. $|P| = 2$) are called *two-player games*. In a two-player game, we write $P = \{+, -\}$ and call $+$ player MAX and $-$ player MIN.

Definition 3.1.17 (Zero-sum)**.** *A two-player game is said to be* zero-sum *if $u_+ + u_- = 0$.*

In other words, a zero-sum game is a game in which the two players have opposite payoffs at every leaf (hence for every outcome): what one gains is equal to what the other loses. Hence, zero-sum games are completely competitive and adversarial.

Zero-sum assumption for maxmin

Notice that in a game, the maxmin value $\underline{v}_i$ of player i depends solely on $\mathcal{U}_i$, and is completely independent of $\mathcal{U}_j$ for $j \neq i$. Hence, when studying the maxmin value of a two-player game, we can always assume without loss of generality that the game is zero-sum.

Furthermore, notice that this means the maxmin strategies of a player are computed without considering the other players' utilities, and *a fortiori* their reasoning or rationality. This is a defining feature of the solution concept of maxmin: it is inherently egocentric and only serves the purpose of guaranteeing one's own best payoff, no matter how other players react.

In this dissertation, we are mainly interested in the solution concept of maxmin for two-player games. Hence, without explicit mention, all games we consider will be two-player zero-sum games.

Boolean games

A particular subclass of zero-sum games is Boolean games.

Definition 3.1.18 (Boolean game)**.** *A zero-sum game is said to be a* Boolean game *if*

$$\forall l \in \mathrm{L}(T), u_+(l) \in \mathbb{B} = \{0, 1\}.$$

Remark. *The values* 0 *and* 1 *are actually inessential; one can choose any two constants: what captures the essence of Boolean games is that there are only two possible payoff values for MAX.*

In a Boolean game, 1 as reward usually signifies a win for player MAX (and a loss for player MIN), and 0 signifies a loss for player MAX (and a win for player MIN). In particular, if the maxmin value for MAX in a game is 1, this means MAX can force a win, no matter how Nature draws at chance nodes and how MIN plays; if the maxmin value for MAX is 0, then MIN can force a loss for MAX.

Boolean game is one of the most studied subclass. Traditionally, Boolean games of no chance and with perfect information are called *combinatorial games*, which is the subject of a vast and rich field of research in both mathematics and computer science; see references about combinatorial game theory in Subsection 2.2.3.

3.2 Information in EFGs

In this section, we briefly study how notions about information[6] are implicitly encoded by the structure of the information sets of the players in an EFG. In particular, we present the notions of perfect recall and multi-agent perfect recall (team games). The treatment here will not be fully rigorous, since our main goal is to give an intuition about these notions. Readers can refer to Maschler et al. (2020) for all missing details.

In the following, we consider G to be an arbitrary EFG of chance.

3.2.1 Perfect recall

Recall that pure strategies are special cases of both mixed strategies and behaviour strategies. On the other hand, the relationship between mixed and behaviour strategies is less straightforward and will be the subject of this subsection.

[6] By "information", we informally refer to what actions in the past a player can observe or retain in memory.

Equivalence of strategies

Consider a profile of mixed and behaviour strategies

$$\varsigma \in \prod_{i=1}^{|P|} (\Sigma_i^{\mathrm{M}} \cup \Sigma_i^{\mathrm{B}}),$$

i.e. for all $i \in P$, ς_i is either a mixed strategy or a behaviour strategy of player i. If each player i implements their strategy ς_i, their drawings of pure strategies (when implementing a mixed strategy) and drawings of actions at every information set (when implementing a behaviour strategy) induce a probability distribution $p(\cdot;\varsigma)\colon V \to [0,1]$ over the vertices of the game tree; for $v \in V$, $p(v;\varsigma)$ is the probability that v will be visited if the players implement the profile ς. Such a probability distribution is called *realisation probability* under the strategy profile ς.

Remark. *We could have defined the expected utility $\mathcal{U}$ for a player $i \in P$ under a profile ς by*

$$\mathcal{U}_i(\varsigma) \coloneqq \sum_{l \in \mathrm{L}(T)} p(l;\varsigma) u_i(l),$$

thereby unifying the notion of expected utility for both mixed and behaviour (hence also pure) profiles.

Definition 3.2.1 (Outcome equivalence)**.** *A mixed strategy σ_i and a behaviour strategy π_i of player i are said to be* (outcome) equivalent *if for all profiles $\varsigma_{-i} \in \prod_{j \neq i} (\Sigma_j^{\mathrm{M}} \cup \Sigma_j^{\mathrm{B}})$ of the players in $P \setminus \{i\}$, the realisation probability is the same under the profile $(\sigma_i, \varsigma_{-i})$ and under the profile (π_i, ς_{-i}):*

$$\forall v \in V, p(v;(\sigma_i,\varsigma_{-i})) = p(v;(\pi_i,\varsigma_{-i})).$$

Mixed strategies without equivalent behaviour strategies

A mixed strategy does not always have an equivalent behaviour strategy. For example, consider the game in Figure 3.5: MAX chooses an action without remembering which one, then chooses a second action. In this game, the mixed strategy $\frac{1}{2}\mathsf{Hh} + \frac{1}{2}\mathsf{Tt}$ (a perfect coordination between MAX's first and second actions) has no equivalent behaviour strategy. Indeed, under this mixed strategy, the leaves l_{Hh} and l_{Tt} will both be reached with probability $\frac{1}{2}$. When using behaviour strategies, to reach l_{Hh} with a non-zero probability, MAX must pick H at the root and h at the children of the root with a non-zero probability; likewise for T and t. But in this case, the leaves l_{Ht} and l_{tH} will be reached with a non-zero probability.

From this example, one can see that in some games, mixed strategies give a player more power than they should have: by implementing a mixed strategy, they can sometimes pick different actions at different vertices in the same information set, based on what they have done in the past. This clearly contradicts our intuition of what an information set should mean; see also the discussion by Koller and Megiddo (1992, Section 2.1).

Behaviour strategies without equivalent mixed strategies

On the other hand, it is also possible that a behaviour strategy does not have any equivalent mixed strategy. For example, consider the game in Figure 3.6: MAX

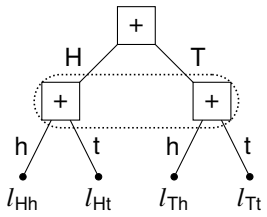


Figure 3.5: A game with a mixed strategy that has no equivalent behaviour strategy. The rewards are omitted for simplicity.

chooses an action without remembering the fact that they have chosen an action, then chooses a second action. In this game, the behaviour strategy $\frac{1}{2}\mathsf{H}+\frac{1}{2}\mathsf{T}$ has no equivalent mixed strategy. Indeed, under this behaviour strategy, the leaf l_{HT} will be reached with probability $\frac{1}{4}$. On the other hand, l_{HT} cannot be reached by any pure strategy (H or T), which means no mixed strategy allows MAX to reach it with non-zero probability. Notice that despite having no equivalent mixed strategy, this behaviour strategy indeed respects the informational constraints on MAX in the game.

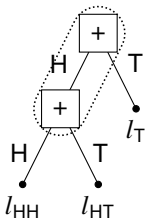


Figure 3.6: A game with a behaviour strategy that has no equivalent mixed strategy. The rewards are omitted for simplicity.

Perfect recall and Kuhn's theorem

We now state necessary and sufficient conditions on a game for all mixed strategies to have an equivalent behaviour strategy, and vice versa. For their proofs, one may refer to Maschler et al. (2020, Chapter 6) or to the original work by Kuhn (1953).

Theorem 3.2.2. *(Maschler et al., 2020, Theorem 6.11) In a game G, every behaviour strategy of player i has an equivalent mixed strategy if and only if no information set of player i is intersected twice by a path starting from the root.*

Remark. *Notice how the game in Figure 3.6 violates this necessary condition: the path from the root to the leaf l_{HH} intersects twice the only information set of MAX in the game.*

For behaviour strategies to be expressive enough so that every mixed strategy has an equivalent behaviour strategy, one needs a stronger condition called perfect recall.

Definition 3.2.3 (Player with perfect recall)**.** *A player i is said to have* perfect recall *(PR) in a game G if every two paths starting from the root and arriving at the same information set of player i pass through the same information sets of player i and in the*

same order, and the same action is chosen by the two paths at each such information set.

Definition 3.2.4 (Game with Perfect Recall)**.** *A game G is said to be with* perfect recall *if all players have perfect recall.*

Notice that perfect information implies perfect recall. Hence, games with perfect information form a subclass of games with perfect recall.

Remark. *The perfect recall of a player implies that no information set of the player is intersected twice by any path from the root: otherwise, there are two paths arriving in such an information set, so that one path only passes through this information set once, while another passes through twice, contradicting the perfect recall of the player.*

Remark. *Notice how the game in Figure 3.5 violates perfect recall: there are two different paths from the root arriving at the information set containing the two children of the root; MAX has taken action* H *in one path, but action* T *in the other.*

Intuitively, perfect recall implies that a player never forgets anything they knew in the past: two paths passing through different sequences of information sets cannot end in the same information set, hence if the player can distinguish two of their decision nodes v and v', then they will not confound a descendant of a v with one of v'. In addition, the player remembers the actions they have chosen in the past, since taking different actions at the same information set will lead to different information sets. Hence, a player with perfect recall always remembers what they saw and did in the past.

Theorem 3.2.5. *(Kuhn, 1953) In a game G, every mixed strategy of player i has an equivalent behaviour strategy if and only if player i has perfect recall.*

3.2.2 Multi-agent perfect recall

The necessary and sufficient condition for every behaviour strategy to have an equivalent mixed strategy in Theorem 3.2.2, which is weaker than perfect recall, has a very natural interpretation; let us first give it a proper name.

Definition 3.2.6 (Multi-agent perfect recall)**.** *A player i is said to have* multi-agent perfect recall *(MA-PR) in a game G if no information set of player i is intersected twice by a path starting from the root.*

Motivation

The term *multi-agent* is motivated by the fact that if the condition in Definition 3.2.6 is satisfied, then player i can be regarded as a team of multiple agents with perfect recall and a shared reward, each agent controlling one of the information sets of player i. Conversely, agents with perfect recall and identical payoffs can be regarded as being controlled by a meta-player who has multi-agent perfect recall.

This multi-agent interpretation of the condition in Definition 3.2.6 dates back to von Stengel and Koller (1997), and is widely adopted in recent research on team games (Basilico et al., 2017; Celli and Gatti, 2018, for instance). In the literature, a *team* is defined to be an inclusion-wise maximal set of players with perfect recall and the same utility function; and recent research focuses on the computational complexity or

algorithms for solution concepts such as TME and TMECor in two-team games with perfect recall;[7] see the references in Chapter 2 on this subject.

As the notion of two-player games with multi-agent perfect recall and the one of two-team games with perfect recall are equivalent, we use these two notions interchangeably throughout this dissertation; we prefer using the first term in order to say that we focus exclusively on two-player games. From now on, to avoid potential confusion, a *player* means a team with zero, one, or more *agents* with perfect recall and a common utility function who take decisions in a completely decentralised manner.[8] Since by definition all agents of a player (i.e. in the same team) have the same payoffs, we only show the rewards for player MAX and player MIN in all our examples of two-player EFGs.

Concurrent actions and non-adaptivity

The notion of multi-agent perfect recall (or team) allows constructing EFGs that model concurrent actions, which are useful for proving complexity results. By *concurrent actions*, we mean situations in which each agent in the same team has to make decisions concurrently and independently, without knowing which action the others have taken. We use the word *concurrent* instead of *simultaneous* to emphasise that the chronological order of the actions of the agents is irrelevant to the game; when modelling such situations with EFG, we can allow agents to take turn in any order.

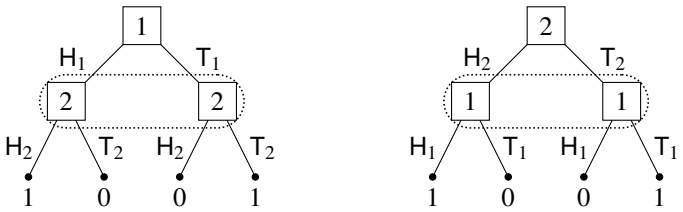


Figure 3.7: Cooperative Matching Pennies.

Example. *Consider the games in Figure 3.7, which we have already seen in Figure 3.5 without showing the payoffs. In these games called cooperative Matching pennies, we have a team MAX of* 2 *agents and a team MIN of* 0 *agent. Notice that on the left, we let agent* 1 *move first, but this is inessential; we may as well let agent* 2 *move first as on the right, which yields another EFG with the same game tree but with different partitions by information sets. These two games model the same situation, in which agent* 1 *and agent* 2 *have to pick between heads and tails concurrently and independently.*

Notice that in these two games, the only pure or behaviour strategies of team MAX to guarantee a win (i.e. a payoff of 1*), are to let both agent* 1 *and agent* 2 *choose heads, or to let them both choose tails. Furthermore, notice that if the two agents have access to a correlation device so that they can coordinate in their mixed strategies, then they can implement a mixture of the two pure strategies above to guarantee a win; this is the problem setting for the solution concept of TMECor (Basilico et al., 2017; Celli and Gatti, 2018).*

Notice how concurrent actions in the example above allow imposing *non-adaptivity*: both agents of MAX must stick to the same answer, otherwise they lose. This is

[7] In our terminology, TME and TMECor are referred to as behaviour maxmin and mixed maxmin, respectively.

[8] In particular, this means we study the notion of (pure or behaviour) TME instead of TMECor.

essentially how multi-agent perfect recall is used to encode difficult (e.g. NP-hard) decision problems.

For instance, consider a 3-colouring game, in which MIN can ask each agent of MAX about their colouring for a vertex.[9] If the two agents of MAX see each other's response (which is roughly equivalent to saying that player MAX has perfect recall instead of just multi-agent perfect recall), then the second agent can pick their response according to the response of the first player. On the other hand, if MIN asks them concurrently and independently (e.g. in the same manner as in the games in Figure 3.7), then since the two agents cannot communicate, they have no way to know whether they are asked about the same vertex or not. In particular, the two agents of MAX have to agree on a colour for each vertex so as to avoid a payoff of 0 in case MIN asks them about the same vertex. But this common 3-colouring also has to be a legal colouring so as to avoid a payoff of 0 in case MIN asks them about vertices connected by an edge.

Remark. *This technique of forcing non-adaptivity is also used to prove that multi-prover interactive proofs are more powerful than single-prover ones (Babai et al., 1991), or that multi-agent planning is computationally more difficult than single-agent planning (Bernstein et al., 2002; Brafman et al., 2013). In this dissertation, we will show similar results concerning multi-agent games with the same technique.*

Best responses for a player with multi-agent perfect recall

Recall that the maxmin value for MAX in a two-player game G with respect to a set of strategies Σ_+ of MAX and a set of profiles Σ_- of MIN is defined to be

$$\underline{v}_+(\Sigma_+, \Sigma_-) \coloneqq \max_{\varsigma_+ \in \Sigma_+} \min_{\varsigma_- \in \Sigma_-} \mathcal{U}_+(\varsigma_+, \varsigma_-).$$

We will show that it makes no difference whether we choose Σ_- to be Σ_-^{P}, or Σ_-^{B}, or Σ_-^{M}, if MIN has multi-agent perfect recall.

Proposition 3.2.7. *Let $\varsigma_+ \in \Sigma_+^{\mathrm{M}} \cup \Sigma_+^{\mathrm{B}}$. Then there exists $s_-^* \in \Sigma_-^{\mathrm{P}}$ with $\mathcal{U}_+(\varsigma_+, s_-^*) = \min_{\sigma_- \in \Sigma_-^{\mathrm{M}}} \mathcal{U}_+(\varsigma_+, \sigma_-)$, i.e. s_-^* is a best response to ς_+ in Σ_-^{M}.*

Proof. Let σ_-^* and s_-^* be two best responses to ς_+ in Σ_-^{M} and Σ_-^{P}, respectively. Then by the definition of $\mathcal{U}_+$ over mixed profiles, we have

$$\mathcal{U}_+(\varsigma_+, \sigma_-^*) = \sum_{s_- \in \Sigma_-^{\mathrm{P}}} \sigma_-^*(s_-) \mathcal{U}_+(\varsigma_+, s_-) \geq \sum_{s_- \in \Sigma_-^{\mathrm{P}}} \sigma_-^*(s_-) \mathcal{U}_+(\varsigma_+, s_-^*) = \mathcal{U}_+(\varsigma_+, s_-^*).$$

Hence, $\mathcal{U}_+(\varsigma_+, \sigma_-^*) \geq \mathcal{U}_+(\varsigma_+, s_-^*)$ holds, that is, $s_-^* \in \Sigma_-^{\mathrm{P}}$ is at least as good a response to ς_+ as σ_-^*; the converse $\mathcal{U}_+(\varsigma_+, \sigma_-^*) \leq \mathcal{U}_+(\varsigma_+, s_-^*)$ follows directly from $\Sigma_-^{\mathrm{P}} \subseteq \Sigma_-^{\mathrm{M}}$ and the definition of a best response in a set of strategies. □

Corollary 3.2.8. *If MIN has multi-agent perfect recall, then for all subsets $\Sigma_+ \subseteq \Sigma_+^{\mathrm{M}} \cup \Sigma_+^{\mathrm{B}}$, we have $\underline{v}_+(\Sigma_+, \Sigma_-^{\mathrm{P}}) = \underline{v}_+(\Sigma_+, \Sigma_-^{\mathrm{B}}) = \underline{v}_+(\Sigma_+, \Sigma_-^{\mathrm{M}})$.*

Proof. Since pure strategies are special cases of behaviour strategies, and since every behaviour strategy of MIN has an equivalent mixed strategy (because MIN has multi-agent perfect recall), we have $\underline{v}_+(\Sigma_+, \Sigma_-^{\mathrm{P}}) \geq \underline{v}_+(\Sigma_+, \Sigma_-^{\mathrm{B}}) \geq \underline{v}_+(\Sigma_+, \Sigma_-^{\mathrm{M}})$, and by Proposition 3.2.7, all inequalities are equalities. □

[9] See the proof of Proposition 4.2.8 for more details.

Part II

Contributions

Chapter 4

Complexity of pure maxmin in extensive-form games

4.1 Introduction

In this chapter, we study the complexity of finding optimal pure strategies in two-player EFGs. This work is motivated by Koller and Megiddo (1992), who establish many positive and negative complexity results concerning maxmin values. However, their study does not systematically cover all relevant subclasses of EFGs. Hence, we extend their study by filling the gaps and strengthening some of their results; in doing so, we provide the first relatively complete landscape for the complexity of pure maxmin.

Common problem setting for the chapter

The games we consider in this chapter will be two-player EFGs. As we have shown in Subsection 3.1.4, we can assume without loss of generality that all games we consider are zero-sum.

We will allow MAX and MIN to have different degrees of imperfect information: perfect information, perfect recall, multi-agent perfect recall (i.e. team games). The only case we do not consider is the one in which a player does not even have multi-agent perfect recall (e.g. the player MAX in Figure 3.6 on page 27). This case is generally referred to as imperfect recall (which, unfortunately, is not precise enough) or *absent-mindedness* (Piccione and Rubinstein, 1997). For the computational complexity of single-player EFGs with absent-mindedness, see the recent work by Gimbert et al. (2020) and Tewolde et al. (2023).

Organisation of the chapter

This chapter is organised as follows.

- In Section 4.2, we investigate the complexity of the decision problem Pure Maxmin, i.e. deciding whether MAX can guarantee a certain payoff by using only pure strategies. Different subclasses of two-player EFGs of chance are considered, parameterised by MAX's and MIN's degree of imperfect information, and the existence of chance.

- After establishing the complete complexity landscape for PURE MAXMIN for EFGs, we pursue in three orthogonal directions.
  - In Section 4.3, we investigate the same problem, but for EFGs that are compactly represented. We propose two formalisms for compactly represented games: compact Boolean games and oracle games. Compact Boolean games aim to generalise minimally QBF and DQBF (proposed by Peterson et al. (2001)) so as to allow MIN to be multi-agent; Oracle games aim to be as general as possible, while respecting some computability criteria, so as to capture tabletop[1] or video games that humans play, which are usually described by game rules.
  - In Section 4.4, we investigate the problem of finding optimal pure strategies for MAX in EFGs when the strategies of MIN are restricted to a small subset (called opponent models) known by MAX.
  - In Section 4.5, we investigate two other variants of PURE MAXMIN for EFGs, in which we decide whether the pure maxmin value is *at most* or *exactly* a given threshold, respectively.
- We then conclude our investigations on this matter by comparing our results with results in the literature on behaviour maxmin, and discussing some possible future work.

4.2 Complexity of EFGs

In this section, we study the complexity of deciding whether the pure maxmin value of a given zero-sum EFG is above a given threshold, when the game tree of the EFG is *explicitly* given in the input.

Definition 4.2.1 (PURE MAXMIN)**.** *Let $\mathcal{G}$ be a class of zero-sum EFGs. Then* PURE MAXMIN($\mathcal{G}$) *is the following decision problem.*

Input *An EFG $G \in \mathcal{G}$ and a rational number m.*

Output *Decide whether $\underline{v}_+(\Sigma_+^{\mathrm{P}}, \Sigma_-^{\mathrm{P}}) \geq m$ holds in G.*

For complexity analyses, to account for the explicit representation of the game, we define the *size* of an instance $\langle G, m\rangle$ of PURE MAXMIN to be $\|G\| + \|m\|$, where for a rational number m, we define $\|m\|$ to be the number of bits in the representation of m, and for an EFG of chance

$$G \coloneqq \left(T, \{+,-\}, (V_i)_{i\in\{0,+,-\}}, (p_v)_{v\in V_0}, u, \left(\langle IS_i^j, A_i^j\rangle\right)_{i\in\{+,-\}}^{j=1,\ldots,k_i}\right),$$

we define

$$\|G\| \coloneqq |T| + \max_{l\in \mathrm{L}(T)} (\|u_+(l)\|) + \max_{v\in V_0, v'\in \mathrm{C}(v)} (\|p_v(v')\|).$$

We will study the complexity of PURE MAXMIN for the classes $\mathcal{G}$ defined by three parameters:

[1] Tabletop games include board games (e.g. Chess, Backgammon, Go), card games (e.g. Bridge, Poker, Hearts), dice games (e.g. Yahtzee), tile-based games (e.g. Mahjong, Dominoes), tabletop role-playing games (e.g. Dungeons & Dragons), strategy games (e.g. wargames), etc.

- MAX's degree of imperfect information: perfect information (PI), perfect recall (PR), or multi-agent perfect recall (MA-PR);
- MIN's degree of imperfect information: PI, PR, or MA-PR;
- the existence of chance nodes: games of no chance, or of chance.

Notice that we are interested in maxmin values with respect to all MIN's pure strategies Σ^{P}_-, which is equivalent to considering all behaviour or mixed strategies of MIN for all subclasses of games we consider, by Corollary 3.2.8.

Moreover, we can further simplify the analysis for games of no chance. Given an EFG G of no chance, let $\mathrm{PI}_{\mathrm{MIN}}(G)$ be the EFG of no chance obtained from G by replacing the set of information sets of MIN by the set of all singleton nodes. Then, in $\mathrm{PI}_{\mathrm{MIN}}(G)$, the game tree, the payoff functions, MAX's information sets, and the set of MAX's pure strategies are the same as in G, but MIN has perfect information.

Lemma 4.2.2. *Let G be an EFG of no chance in which MIN has multi-agent perfect recall. Then the pure maxmin value for MAX in G is the same as the pure maxmin value for MAX in* $\mathrm{PI}_{\mathrm{MIN}}(G)$.

Proof. First, since MIN has no fewer strategies in $\mathrm{PI}_{\mathrm{MIN}}(G)$ than in G (since in $\mathrm{PI}_{\mathrm{MIN}}(G)$ MIN can choose different actions at nodes in the same information set of G), the maxmin value for MAX cannot be strictly higher in $\mathrm{PI}_{\mathrm{MIN}}(G)$ than in G.

Conversely, let $s_+ \in \Sigma^{\mathrm{P}}_+, s'_- \in \Sigma^{\mathrm{P}}_-$ be pure strategies for MAX and MIN, respectively, in $\mathrm{PI}_{\mathrm{MIN}}(G)$. The pure strategy profile (s_+, s'_-) in $\mathrm{PI}_{\mathrm{MIN}}(G)$ uniquely determines a playout of the game. Since MIN has MA-PR in G, this path intersects every information set of MIN in G at most once. Hence, one can define a pure strategy s_- for MIN in G such that at every decision node of MIN in the path, MIN takes the same action as in s'_-.[2] By construction, the playout is the same under the pure strategy profile (s_+, s_-) in G as under (s_+, s'_-) in $\mathrm{PI}_{\mathrm{MIN}}(G)$, hence the utility for MAX is the same. It follows that the pure maxmin value for MAX cannot be strictly higher in G than in $\mathrm{PI}_{\mathrm{MIN}}(G)$, which concludes the proof. □

Since $\mathrm{PI}_{\mathrm{MIN}}(G)$ can clearly be built in polynomial time from G, it follows that for every fixed degree of imperfect information of MAX, the degree of imperfect information of MIN (PI, PR, or MA-PR) does not influence the complexity of Pure Maxmin for EFGs of no chance.

4.2.1 Summary of results

The complexity of Pure Maxmin($\mathcal{G}$) is summarised in Table 4.1. By definition,[3] the complexity of each case is increasingly monotone in all three parameters: MAX's degree of imperfect information (in this order: PI, PR, MA-PR), MIN's degree of imperfect information, and the existence of chance nodes (in this order: no chance, chance). Hence, Table 4.1 only gives the references for the key hardness ("h") and membership ("m") results; the other results can be deduced using monotonicity. Note that results written in bold font are new from our work; the others can be directly deduced from the literature.

[2] Concretely, at an information set IS_- of MIN in G, if IS_- intersects exactly once the playout in $\mathrm{PI}_{\mathrm{MIN}}(G)$ under (s_+, s'_-), then $s_-(IS_-)$ is defined to be the action picked by MIN under s'_- in $\mathrm{PI}_{\mathrm{MIN}}(G)$ at the unique intersection of IS_- and the playout; otherwise, if IS_- does not intersect the playout, then $s_-(IS_-)$ can be defined to be any available action at IS_-.

[3] Recall that, for instance, a game of no chance is a game of chance; a game with perfect information is a game with perfect recall.

MAX \ MIN	No chance	Chance		
	PI/PR/MA-PR	PI	PR	MA-PR
PI	P	P [m: 4.2.9]	**NP-c [h: 4.2.10]**	**Σ_2^P-c [h: 4.2.12]**
PR	**P [m: 4.2.4]**	NP-c [h: 4.2.11]	NP-c	**Σ_2^P-c**
MA-PR	NP-c [h: 4.2.8]	NP-c	NP-c [m: 4.2.3]	Σ_2^P-c [m: 4.2.14]

Table 4.1: Complexity of Pure Maxmin. All hardness results hold even under the restriction to Boolean games with at most 2 agents for both MAX and MIN.

4.2.2 EFG of no chance

Koller and Megiddo (1992, Section 3.3) show that if MIN has perfect recall or there is no chance, then for all pure strategies of MAX, a best response for MIN can be computed in linear time. This gives an upper bound of NP for most cases in Table 4.1.

Proposition 4.2.3. Pure Maxmin *is in* NP *for EFGs of no chance in which MAX and MIN have MA-PR, and for EFGs of chance in which MAX has MA-PR and MIN has PR.*

Proof. One can guess a pure strategy s_+ for MAX, then verify that it yields an expected payoff no less than the given threshold, by computing a best response to s_+ for MIN, which can be done in linear time when MIN has PR or there is no chance (Koller and Megiddo, 1992, Section 3.3). □

When MAX has perfect recall, we show that the problem is actually in P.

Proposition 4.2.4. Pure Maxmin *is decidable in linear time, and* a fortiori *is in* P*, for EFGs of no chance in which MAX has PR and MIN has MA-PR.*

For the proof, we introduce the notion of reachability in games of no chance.

Definition 4.2.5 (Reachability)**.** *Let $s_+ \in \Sigma_+^P$ be a pure strategy of MAX in an EFG of no chance. A vertex $v \in V$ is said to be* reachable *under s_+ if there is a pure strategy of MIN $s_- \in \Sigma_-^P$ such that v is in the playout under the profile (s_+, s_-). An information set (of MAX or MIN) is said to be* reachable *under s_+ if at least one vertex in it is reachable under s_+. The* reachability *of a vertex or an information set under a pure strategy of MIN is defined similarly.*

For every vertex $v \in V$ in the game tree, we define $\Sigma_{+,v}^P \subseteq \Sigma_+^P$ to be the set of pure strategies of MAX under which v is reachable (in particular, $\Sigma_{+,r}^P = \Sigma_+^P$); The set $\Sigma_{-,v}^P$ is defined similarly.

Proof of Proposition 4.2.4. Let us consider a two-player game of no chance in which MAX has perfect recall and MIN has multi-agent perfect recall. We will provide a linear-time algorithm to compute the maxmin value of such a game.

We recursively define the function $ev : V \to \mathbb{R}$ on all vertices of the game tree:

- If v is a leaf, then the value of this node is given by the utility function of MAX: $ev(v) \coloneqq u_+(v)$.
- If v is a decision node of MIN, then the value of this node is the minimum of the value of its children: $ev(v) \coloneqq \min_{v' \in C(v)} ev(v')$.

- Otherwise, v is a decision node of MAX. Let IS_+ be the information set containing v. Then the value of this node is

$$ev(v) \coloneqq \max_{a \in A_+} \min_{v' \in IS_+} ev(a(v')), \tag{4.1}$$

where $a(v') \coloneqq a \cap \mathrm{C}(v')$ denotes the vertex reached by taking action a at node v'.

The function ev is well-defined since MAX's information sets form a forest due to MAX's perfect recall (Koller and Megiddo, 1992, Proposition 3.1). In addition, ev has the same value for vertices in the same information set of MAX: in (4.1), the minimum ranges over all successors of the information set and the maximum ranges over all available actions at that information set. Both ranges depend solely on the information set itself, and are therefore the same for all vertices in it.

Our goal is to show that the maxmin value is given by the value of the root:

$$\underline{v}_+ \coloneqq \max_{s_+ \in \Sigma^{\mathrm{P}}_+} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-) = ev(r). \tag{4.2}$$

Since $ev(r)$ is computable in linear time using its recursive definition, (4.2) implies that the maxmin value is computable in linear time, thus proving Proposition 4.2.4.

We first show that $ev(r)$ is an upper bound of $\underline{v}_+$ with the help of a lemma proven in Appendix B.1.1:

Lemma 4.2.6. *For every vertex $v \in V$, it holds that*

$$ev(v) \geq \max_{s_+ \in \Sigma^{\mathrm{P}}_{+,v}} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-). \tag{4.3}$$

This lemma gives one interpretation of the value $ev(v)$: it is the upper bound of the guaranteed payoff of any pure strategy of MAX's under which v can be reached. Applying (4.3) to the root and noticing $\Sigma^{\mathrm{P}}_{+,r} = \Sigma^{\mathrm{P}}_+$, we obtain

$$ev(r) \geq \max_{s_+ \in \Sigma^{\mathrm{P}}_{+,r}} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-) = \max_{s_+ \in \Sigma^{\mathrm{P}}_+} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-) = \underline{v}_+. \tag{4.4}$$

To complete the proof of Proposition 4.2.4, we still need to show that the value $ev(r)$ can actually be achieved, i.e. there exists a pure strategy s^*_+ of MAX such that $\min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s^*_+, s_-) = ev(r)$.

For this, first notice that ev implicitly defines a non-empty set of pure strategies of MAX $\Sigma^{\mathrm{P}}_{+,ev} \subseteq \Sigma^{\mathrm{P}}_+$ in the following sense: at each information set IS_+ of MAX, a strategy in $\Sigma^{\mathrm{P}}_{+,ev}$ assigns an action in A_+ such that the maximum in (4.1) is achieved. Let $s^*_+ \in \Sigma^{\mathrm{P}}_{+,ev}$ be an arbitrary pure strategy defined by ev. We can show that s^*_+ achieves the value $ev(r)$ with the help of another lemma proven in Appendix B.1.1:

Lemma 4.2.7. *Let $s^*_+ \in \Sigma^{\mathrm{P}}_{+,ev}$. For every vertex v reachable under s^*_+, it holds that*

$$ev(v) \leq \min_{s_- \in \Sigma^{\mathrm{P}}_{-,v}} \mathcal{U}_+(s^*_+, s_-). \tag{4.5}$$

This lemma gives another interpretation of the value $ev(v)$: it is the lower bound of the payoff of MAX's best strategies when MIN only plays pure strategies under which v can be reached. Applying (4.5) to the root and noticing $\Sigma^{\mathrm{P}}_{-,r} = \Sigma^{\mathrm{P}}_-$, we obtain

$$ev(r) \leq \min_{s_- \in \Sigma^{\mathrm{P}}_{-,r}} \mathcal{U}_+(s^*_+, s_-) = \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s^*_+, s_-) \leq \max_{s_+ \in \Sigma^{\mathrm{P}}_+} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-) = \underline{v}_+. \tag{4.6}$$

Combining (4.4) and (4.6), we have

$$\underline{v}_+ \leq ev(r) \leq \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s^*_+, s_-) \leq \underline{v}_+,$$

hence all inequalities are equalities: $\underline{v}_+ = ev(r) = \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s^*_+, s_-)$ for all pure strategies of MAX s^*_+ in $\Sigma^{\mathrm{P}}_{+,ev}$. In particular, maxmin strategies of MAX can be found by computing the function ev. □

P-membership in Proposition 4.2.4 holds if MAX has PR, which is the best we can do: if MAX only has MA-PR, then the problem becomes NP-hard.

Proposition 4.2.8. Pure Maxmin *is* NP*-hard for EFGs of no chance in which MAX has MA-PR. The result holds even under the restriction to* 2 *agents for MAX and to Boolean games.*

Proof. We give a reduction from the NP-complete problem 3-Colouring, which is defined as follows:

Input A non-directed graph (V, E).

Output Decide whether the graph has a legal 3-colouring, which is a mapping $C : V \to \{1, 2, 3\}$ such that for all $(v_1, v_2) \in E$, $C(v_1) \neq C(v_2)$.

Let (V, E) be an instance of 3-Colouring. We use the ideas of concurrent actions and non-adaptivity from Subsection 3.2.2 to build the following game.

Players MAX with multi-agent perfect recall of 2 agents, labelled by 1 and 2; MIN with perfect information.

Game tree At the root, MIN chooses two vertices $v_1, v_2 \in V$, and concurrently and independently shows v_i to agent i of MAX. Then each agent of MAX concurrently and independently chooses a colour ($c_1, c_2 \in \{1, 2, 3\}$), and the game ends.[4]

Payoffs for MAX MAX receives 0 if either of the following conditions is satisfied:

- $v_1 = v_2$ but $c_1 \neq c_2$;
- $(v_1, v_2) \in E$ but $c_1 = c_2$.

Otherwise, MAX receives 1.

Maxmin The threshold of maxmin value is 1.

The construction is polynomial-time in the input (V, E). Indeed, the game tree is of size $O(|V|^2)$. In addition, the construction yields a Boolean EFG of no chance in which MAX has multi-agent perfect recall and 2 agents, and MIN has perfect information.

We now show that the pure maxmin value of the game is 1 if (V, E) has a legal 3-colouring, and 0 otherwise. Notice first that the set of pure strategies of each agent of MAX is in bijection with the set of 3-colourings of the graph.

[4]Concretely, we may construct the game tree in the following way: MIN first picks v_1. Each choice leads to a different information set for agent 1 (thus agent 1 knows v_1), who chooses a colour c_1. Each choice of v_1 and c_1 leads to a different information set for MIN (thus MIN has perfect information), who chooses a vertex v_2. Each choice of v_2 leads to a different information set of agent 2, which is the same for all choices of v_1 and c_1 (thus agent 2 is unaware of v_1 and c_1, but knows v_2). Finally, agent 2 chooses a colour c_2.

- Suppose that (V, E) has a legal 3-colouring $C^* : V \to \{1, 2, 3\}$. Then, if each agent of MAX plays C^* as their strategy, MAX always obtains a payoff of 1 since neither condition for a zero payoff can be met. Hence, the maxmin value of the game is 1 since 1 is the largest possible payoff of the game.

- Conversely, suppose that (V, E) does not have a legal 3-colouring. We can show that every pure strategy of MAX gets a payoff of 0 against MIN's best response.

  - If the agents of MAX play two different pure strategies (i.e. they pick two different 3-colourings), their strategies differ on at least one vertex, say $v \in V$. Then MIN can play $v_1 = v_2 = v$ so that MAX gets a payoff of 0.
  - If the agents of MAX pick a common 3-colouring $C : V \to \{1, 2, 3\}$, then since C cannot be a legal 3-colouring, there is $(v_1^*, v_2^*) \in E$ such that $C(v_1^*) = C(v_2^*)$. Hence, MIN can play $v_1 = v_1^*$ and $v_2 = v_2^*$ so that MAX gets a payoff of 0.

As a conclusion, the pure maxmin value of the game is 1 if the graph has a legal 3-colouring; if no such colouring exists, then the pure maxmin value is 0. □

Remark. *Koller and Megiddo (1992, Proposition 2.6) prove a similar result using a reduction from* 3-SAT. *We provide a simpler reduction, which suits better our needs to prove several other results in this dissertation, and also strengthens their result to* 2 *agents (instead of* 3*).*

Remark. *The contrast between* NP*-hardness for EFGs of no chance in which MAX has MA-PR and MIN has PI (as in Proposition 4.2.8) and membership in* P *for EFGs of no chance in which MAX has PI and MIN has MA-PR (as in Proposition 4.2.4) shows that there is an asymmetry between MAX and MIN in the computation of maxmin values, when MAX can only use pure strategies. This is in contrast with the symmetry between MAX and MIN implied by the minimax theorem when MAX can use mixed strategies:*

$$\max_{\sigma_+ \in \Sigma_+^{\mathrm{M}}} \min_{\sigma_- \in \Sigma_-^{\mathrm{M}}} \mathcal{U}_+(\sigma_+, \sigma_-) = \min_{\sigma_- \in \Sigma_-^{\mathrm{M}}} \max_{\sigma_+ \in \Sigma_+^{\mathrm{M}}} \mathcal{U}_+(\sigma_+, \sigma_-). \tag{4.7}$$

For pure strategies, in general, we only have

$$\max_{s_+ \in \Sigma_+^{\mathrm{P}}} \min_{\sigma_- \in \Sigma_-^{\mathrm{M}}} \mathcal{U}_+(s_+, \sigma_-) \leq \min_{\sigma_- \in \Sigma_-^{\mathrm{M}}} \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \mathcal{U}_+(s_+, \sigma_-),$$

and the inequality is strict, for instance, in the game Matching Pennies (Figure 3.3).

4.2.3 EFGs of chance

In games with perfect information for both players, the restriction to pure strategies is inessential, as pure strategies give as high a payoff as mixed strategies. The minimax algorithm and alpha-beta search (Knuth and Moore, 1975) indeed compute the maxmin value in linear time for EFGs of no chance and with perfect information. With minor modifications, these algorithms also work when there are chance nodes (Ballard, 1983).

Proposition 4.2.9. Pure Maxmin *is decidable in linear time, and* a fortiori *is in* P*, for EFGs of chance in which both MAX and MIN have PI.*

However, it becomes hard when at least one player does not have perfect information.[5]

Proposition 4.2.10. PURE MAXMIN *is* NP*-hard for EFGs of chance in which MAX has PI and MIN has PR. The result holds even under the restriction to Boolean games.*

Proof. We give again a reduction from 3-COLOURING. Given an instance (V, E) of 3-COLOURING, we build the following Boolean EFG.

Players Nature; MAX with perfect information; MIN with perfect recall.

Game tree The game proceeds as follows:

1. At the root, Nature chooses uniformly at random a vertex $v \in V$ and shows it to MAX (i.e. each choice of v leads to a different information set of MAX).
2. After observing v, MAX chooses a colour $c_+ \in \{1, 2, 3\}$.
3. Without observing v nor c_+ (i.e. all choices of v and c_+ lead to the same information set of MIN), MIN chooses an edge $e \in E$ and a colour $c_- \in \{1, 2, 3\}$.

Payoffs for MAX MAX receives 0 if v is part of the edge e and $c_+ = c_-$. Otherwise, MAX receives 1.

Maxmin The threshold of maxmin value is $1 - 1/|V|$.

The construction is polynomial-time in the input (V, E). Indeed, the game tree is of size $O(|V||E|)$. In addition, the construction yields a Boolean EFG of chance in which MAX has perfect information and MIN has perfect recall.

We now show that if (V, E) has a legal 3-colouring, then the maxmin value of the game is at least $1 - 1/|V|$; conversely, if the graph is not 3-colourable, then the maxmin value of the game is at most $1 - 2/|V|$. Notice first that the set of pure strategies of each agent of MAX is in bijection with the set of 3-colourings of the graph.

- Suppose that (V, E) has a legal 3-colouring C^* and that MAX plays C^*. Consider MIN's strategy $(v', v'', c_-) \in E \times \{1, 2, 3\}$. Since C^* is a legal 3-colouring, $C^*(v') \neq C^*(v'')$, hence MAX assigns the colour c_- to at most one of v' and v''. Whenever Nature does not pick this vertex as v, MAX gets a payoff of 1. It follows that MAX gets a payoff of 1 with probability at least $1 - 1/|V|$.

- Conversely, suppose that (V, E) does not have a legal 3-colouring. Let $C : V \to \{1, 2, 3\}$ be a pure strategy of MAX. Then there is $(v', v'') \in E$ and a colour c such that $C(v') = C(v'') = c$. If MIN plays the strategy $e = (v', v'')$ and $c_- = c$, then MAX gets 0 with probability $2/|V|$ (i.e. whenever Nature chooses v' or v'' as v). Hence, MAX's expected payoff is at most $1 - 2/|V|$.

As a conclusion, the pure maxmin value of the game is at least $1 - 1/|V|$ if the graph has a legal 3-colouring; if no such colouring exists, then the pure maxmin value is at most $1 - 2/|V|$. □

Proposition 4.2.11. PURE MAXMIN *is* NP*-hard for EFGs of chance in which MAX has PR and MIN has PI. The result holds even under the restriction to Boolean games.*

[5] In contrast, computing optimal behaviour strategies for MAX is still in P (Koller and Megiddo, 1992, Section 3), even when both players only have perfect recall.

Proof. We adapt the reduction in the proof of Proposition 4.2.10 by changing the game tree as follows.

1. At the root, MIN chooses an edge $e \in E$ and a colour $c_- \in \{1, 2, 3\}$.
2. Then Nature chooses uniformly at random a vertex $v \in V$.
3. After knowing v but without observing e nor c_- (i.e. each choice of v leads to a unique information set of MAX, which is the same for all choices of e and c_-), MAX chooses a colour $c_+ \in \{1, 2, 3\}$.
4. The payoffs for MAX are as defined in the proof of Proposition 4.2.10.

The game is now a Boolean game of chance in which MAX has PR and MIN has PI. The reasoning is exactly the same as in the proof of Proposition 4.2.10, and the same conclusion holds: the pure maxmin value of the game is at least $1 - 1/|V|$ if the graph has a legal 3-colouring, and otherwise it is at most $1 - 2/|V|$. □

Remark. *A similar result is proven by Frank and Basin (2001, Section 6) for EFGs with one-sided incomplete information by using a reduction from the* NP*-complete problem* Clique. *See the proof of Proposition 5.3.3.*

We now consider games of chance in which MIN only has multi-agent perfect recall. Proposition 2.10 by Koller and Megiddo (1992) implies that if MAX also only has multi-agent perfect recall, then Pure Maxmin is Σ_2^{P}-hard. We use a different reduction to strengthen their result and show that actually Maxmin remains Σ_2^{P}-hard even if MAX has perfect information.

Proposition 4.2.12. Pure Maxmin *is Σ_2^{P}-hard for EFGs of chance in which MAX has PI and MIN has MA-PR. The result holds even under the restriction to* 2 *agents for MIN and to Boolean games.*

For this proof, we will consider a reduction from a variant of the domino tiling problem. We use the following definition for tiling (Grädel, 1990; Goldsmith and Mundhenk, 2007).

Definition 4.2.13 (Legal tiling)**.** *Let D be a finite set, $H, V \subseteq D \times D$ be two binary relations, and let $S = \{1, \dots, m\} \times \{1, \dots, m\}$ with $m \geq 2$. A* tiling *of S is a mapping $\tau : S \to D$. Such a tiling is called a* legal tiling *(with respect to H and V) if:*

- *Every pair of two horizontally adjacent tiles is in H:*

$$\forall r \in \{1, \dots, m-1\}, \forall c \in \{1, \dots, m\}, (\tau(r, c), \tau(r+1, c)) \in H;$$

- *Every pair of two vertically adjacent tiles is in V:*

$$\forall r \in \{1, \dots, m\}, \forall c \in \{1, \dots, m-1\}, (\tau(r, c), \tau(r, c+1)) \in V.$$

A legal tiling of a board of any shape is defined similarly.

The decision problem Tiling consists in, given $(D, H, V, 1^m)$,[6] deciding whether there is a legal tiling of the square board of size m. Tiling is known to be NP-complete

[6] 1^m is the unary expression of m.

(van Emde Boas, 1997).[7] Variants of tiling problems provide plenty of complete problems for different complexity classes (Schwarzentruber, 2019). For example, the following variant called TILING EXTENSION is Σ_2^{P}-complete (Schaefer and Umans, 2002).

Input A finite set D, two binary relations $H, V \subseteq D \times D$, a natural number m expressed in unary.

Output Decide whether there is a legal tiling of the first row of $S = \{1, \dots, m\} \times \{1, \dots, m\}$ that cannot be extended to a legal tiling of S.

In the following, we refer to a legal tiling of the first row that cannot be extended to a legal tiling of the whole board as a *non-extendable legal tiling* of the first row.

Proof of Proposition 4.2.12. We give a reduction from TILING EXTENSION. Given an instance $(D, H, V, 1^m)$ of TILING EXTENSION, we build a game in which MAX wins if they can choose a legal tiling of $\{1\} \times \{1, \dots, m\}$, such that whatever tiling τ of the whole board S MIN chooses, either τ is not legal, or it does not extend MAX's tiling.

Precisely, we build the following game.

Players Nature; MAX with perfect information; MIN with multi-agent perfect recall of 2 agents, labelled by 1 and 2.

Game tree The game begins with a chance node and proceeds as follows:

1. At the root, Nature chooses uniformly at random a column $c \in \{1, \dots, m\}$, and only shows it to MAX.
2. MAX chooses a tile $d \in D$.
3. Nature then chooses uniformly at random two positions $(r_1, c_1), (r_2, c_2) \in S$, then concurrently and independently shows agent i of MIN the position (r_i, c_i).
4. Without observing c nor d, each agent of MIN concurrently and independently chooses a tile ($d_1, d_2 \in D$), and the game ends.

Payoffs for MAX The payoff for MAX at the leaf induced by c, d, r_1, c_1, d_1, r_2, c_2, d_2 is defined to be:

1. $2m^5$, if $(r_1, c_1) = (r_2, c_2)$ but $d_1 \neq d_2$ (MAX wins if agent 1 and agent 2 of MIN are inconsistent);
2. otherwise, $2m^4$, if $(r_1, c_1) = (1, c)$ but $d_1 \neq d$, or if $(r_2, c_2) = (1, c)$ but $d_2 \neq d$ (MAX wins if the tiling of any agent of MIN does not extend theirs);
3. otherwise, $-m^4$ if $r_1 = r_2 = 1$ and $c_2 = c_1 + 1$ but $(d_1, d_2) \notin H$ (MAX loses if their tiling is illegal, which is verified with MIN's choices);
4. otherwise, 1 if $r_1 = r_2 \neq 1$ and $c_2 = c_1 + 1$ but $(d_1, d_2) \notin H$ (MAX wins if MIN's tiling is illegal with respect to H in rows other than the first one);
5. otherwise, 1 if $c_1 = c_2$ and $r_2 = r_1 + 1$ but $(d_1, d_2) \notin V$ (MAX wins if MIN's tiling is illegal with respect to V);
6. otherwise, 0.

Maxmin The threshold of maxmin value is $1/m^4$.

[7] Remark that 3-COLOURING and TILING have very similar structure; in the previous section, all proofs based on 3-COLOURING can be easily adapted to be based on TILING.

The construction is polynomial-time in the input $(D, H, V, 1^m)$. Indeed, the game tree is of size $O(m^5|D|^3)$, and the computation of the payoffs of the leaves is polynomial-time. In addition, the construction yields an EFG of chance in which MAX has perfect information and MIN has multi-agent perfect recall of 2 agents.

We will show that the pure maxmin value of this game is at least $1/m^4$ if S has a non-extendable legal tiling of the first row (and at most 0 otherwise). First, observe that the pure strategies of MAX are in bijection with the tilings of the first of row of S. Similarly, the pure strategies of each agent of MIN are in bijection with the tilings of S.

Furthermore, observe that MAX gets a strictly negative payoff only when $r_1 = r_2 = 1$ and $c_2 = c_1 + 1$ (and $(d_1, d_2) \notin H$), which happens with probability at most $1/m^3$; hence MAX cannot get an expected payoff smaller than $(1/m^3) * (-m^4) = -m$ from the corresponding leaves.

$\Longrightarrow$ Suppose first that there is a non-extendable legal tiling of the first row $\tau_0 : \{1, \ldots, m\} \to D$. Let (τ_1, τ_2) denote an arbitrary pure strategy of MIN, where τ_1 and τ_2 are tilings of S.

- If $\tau_1 \neq \tau_2$ holds, then with probability at least $(1/m^2)^2 = 1/m^4$, Nature chooses $(r_1, c_1) = (r_2, c_2)$ over which τ_1, τ_2 differ, and MAX gets $2m^5$. Since the smallest expected payoff they can get from negative leaves is $-m$, the expected payoff of MAX against (τ_1, τ_2) is at least $1/m^4 \times 2m^5 - m \geq 1/m^4$.

- Now assume $\tau_1 = \tau_2$, but τ_1 is not an extension of τ_0. Then, with probability at least $1/m^3$, Nature chooses c, $r_1 = 1$ and $c_1 = c$ such that τ_1 differs from τ_0 on $(1, c)$, and MAX wins $2m^4$. As above, it follows that the expected payoff of MAX against (τ_1, τ_2) is at least $1/m^3 \times 2m^4 - m \geq 1/m^4$.

- Finally, if $\tau_1 = \tau_2$ holds and τ_1 is an extension of τ_0, then τ_1 is illegal by definition of τ_0. Then with probability at least $1/m^4$, Nature chooses $(r_1, c_1), (r_2, c_2)$ to witness the illegality, and MAX gets 1. It follows again that the expected payoff of MAX against (τ_1, τ_2) is at least $1/m^4$ (observe that in this case, MAX cannot get a negative payoff under other choices by Nature: τ_1 and τ_2 are legal on the first row since they extend τ_0).

Hence, playing τ_0 guarantees a payoff of at least $1/m^4$ for MAX, which means the maxmin value of the game is at least $1/m^4$.

$\Longleftarrow$ Conversely, suppose that the maxmin value of the game is at least $1/m^4$, and let τ_0 be a pure strategy of MAX achieving this value.

- We claim that τ_0 is legal. Indeed, otherwise MIN can play the strategy (τ, τ), where $\tau : S \to D$ is an arbitrary extension of τ_0. Under this profile, with probability at least $1/m^4$ Nature chooses $r_1 = r_2 = 1$ and $c_1, c_2 = c_1 + 1$ witnessing the illegality of τ on the first row, resulting in a negative payoff of $-m^4$ for MAX. Moreover, under this profile MAX cannot get a payoff of $2m^5$ nor $2m^4$, so they get at most 1 from all other leaves. It follows that the maxmin value is at most $1/m^4 \times (-m^4) + 1 \leq 0$, contradicting the assumption.

- We also claim that any extension $\tau : S \to D$ of τ_0 is illegal. Indeed, otherwise MIN can play the strategy (τ, τ), which yields a payoff of 0 for MAX at all leaves, again contradicting the assumption.

Hence, τ_0 is a non-extendable legal tiling of the first row. In addition, by the argument above, we see that if no such τ_0 exists, then MAX gets at most 0 against MIN's best responses.

To show that this result still holds under the restriction to Boolean games, we can use the gadgets in Lemma B.1.5 to compile all integers payoffs in the construction above into Boolean ones. □

We conclude this section by showing the Σ_2^{P}-membership for the most general setting.

Proposition 4.2.14. Pure Maxmin *is in* Σ_2^{P} *for EFGs of chance in which MAX and MIN have MA-PR.*

Proof. One can first guess a pure strategy for MAX, then verify that for each pure strategy of MIN, it yields an expected payoff no less than the threshold. Since each verification takes linear time, the problem is in Σ_2^{P}. □

4.3 Complexity of compactly represented games

In this section, we study the complexity of Pure Maxmin for EFGs represented in compact form. As it turns out, the complexity of Pure Maxmin is very robust to the exact compact representation chosen; hence we formulate hardness results for a very restricted class of representations, and membership results for a very general one. We first introduce these representations and show that they encompass natural representations, then we give the complexity results, which, as it happens, parallel the results for non-compact EFGs.

4.3.1 Compact representations of games

For most tabletop and video games, their game tree is rarely defined explicitly, but rather implicitly by their game rules; such rules allow computing the decision maker, the children, the payoffs, etc., at a given node of the game tree, which can therefore be generated online. In other words, we may say that the trees of these games are represented compactly by their game rules.

Typically, for such a compact representation of a game, the corresponding game tree is exponentially larger, or more. In the following, we introduce two formalisms to capture the intuition of what a *compactly represented game* means.

Compact Boolean games

We first introduce a minimal formalism for compactly represented games, which will be used to prove all hardness results in this section. Our formalism is inspired by *quantified Boolean formulae* (QBF). A QBF is a formula of the form

$$\mathrm{Q}_1\, x_1\, \mathrm{Q}_2\, x_2 \cdots \mathrm{Q}_n\, x_n\, \varphi(x_1, x_2, \ldots, x_n),$$

where for all i, x_i is a Boolean variable (with values in $\mathbb{B} = \{0, 1\}$) and $\mathrm{Q}_i \in \{\forall, \exists\}$,[8] and φ is a plain (unquantified) Boolean formula. A QBF can be regarded as a game of no chance with perfect information in which MAX and MIN take turns to pick a value

[8]The quantifiers $\forall$ and $\exists$ have their standard meaning of "for all" and "exists".

for each variable: MAX is responsible for variables with the existential quantifier $\exists$ and MIN for the universal quantifier $\forall$, and MAX's goal is to render the formula φ true.

Peterson et al. (2001) propose a generalisation of QBF called *dependency quantified Boolean formula* (DQBF), which allows dependencies for existential variables. For example, in the DQBF

$$\forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2)\, \varphi(x_1, y_1, x_2, y_2),$$

the value of y_1 can only depend on the value of x_1, and the value of y_2 only on x_2. Hence, a DQBF represents a game of no chance in which MAX only has multi-agent perfect recall. In the example above, y_1 and y_2 can be regarded as being controlled by two agents of MAX. Notice that DQBF is indeed a generalisation of QBF, since every QBF of size n can be transformed into an equivalent DQBF of size $O(n^2)$ by allowing each existential variable to depend on all previous variables.

As we intend to model games of chance with multi-agent perfect recall for both MAX and MIN, we introduce a generalisation of DQBF that we call *compact Boolean game* (CBG), which can be written into the following form to mimic DQBF:

$$\mathrm{P}_1\, x_1(D_1)\, \mathrm{P}_2\, x_2(D_2) \cdots \mathrm{P}_n\, x_n(D_n)\, \varphi_+(x_1, x_2, \ldots, x_n),$$

where P_j is the owner of x_j, D_j is the set of variables on which x_j depends, and φ_+ is an unquantified Boolean circuit. More formally:

Definition 4.3.1 (Compact Boolean game)**.** *A* compact Boolean game *(CBG) is a tuple $\gamma = \langle X, \mathrm{P}, \mathrm{D}, \varphi_+ \rangle$, with X an ordered list of variables, $\mathrm{P} : X \to \{0, +, -\}$[9] an owner function, $\mathrm{D} : X \to \mathcal{P}(X)$ a dependency function such that for all $x \in X$, $\mathrm{D}(x)$ only contains variables that precede x in the ordering and $\mathrm{D}(x) = \emptyset$ if $\mathrm{P}(x) = 0$, and φ_+ a Boolean circuit with inputs in X and one Boolean output.*

A CBG is said to be of no chance*, if it does not have* chance variables*, that is, variables owned by Nature.*

Intuitively, a CBG represents the Boolean EFG of chance in which for each x_j in turn, player $\mathrm{P}(x_j)$ chooses a Boolean value after observing only the values chosen for the variables in $\mathrm{D}(x_j)$ (and Nature chooses uniformly at random). After all variables have been played, MAX receives the payoff given by the Boolean output of the circuit $\varphi_+(x_1, x_2, \ldots, x_n)$ (and MIN receives the opposite). In particular, a node of the game is given by a variable index j and an assignment to $x_1, \ldots, x_{j-1}$.

To formally define this EFG defined by a CBG, we identify an assignment μ of a set of variables X to the set of variables assigned true, and accordingly write $\mathcal{P}(X)$ for the set of all assignments of X.

Definition 4.3.2 (CBG EFG)**.** *The Boolean EFG of chance defined by a CBG $\gamma = \langle X, \mathrm{P}, \mathrm{D}, \varphi_+ \rangle$, with $X = (x_1, \ldots, x_n)$, is defined to be*

$$G(\gamma) \coloneqq \left(T, \{+, -\}, (V_i)_{i \in \{0,+,-\}}, (p_v)_{v \in V_0}, u, \left(\langle IS_i^k, A_i^k \rangle\right)_{i \in \{+,-\}}^{k}\right),$$

where

- $T = (V, E, r)$ *with* $V \coloneqq \{(j, \mu) \mid 1 \leq j \leq n + 1, \mu \in \mathcal{P}(\{x_1, \ldots, x_{j-1}\})\}$, $E \coloneqq \{(v, v') \in V^2 \mid v = (j, \mu), v' = (j + 1, \mu)$ *or* $(j + 1, \mu \cup \{x_j\})\}$, $r \coloneqq (1, \emptyset)$*;*

[9] Recall that 0 represents the Nature.

- $V_i \coloneqq \{(j,\mu) \in V \mid \mathrm{P}(x_j) = i\}$;
- *For every* $1 \leq j \leq n$ *such that* $\mathrm{P}(x_j) = 0$,

$$p_{(j,\mu)}((j+1,\mu)) = p_{(j,\mu)}((j+1,\mu \cup \{x_j\})) = 1/2$$

  for all $\mu \in \mathcal{P}(\{x_1,\ldots,x_{j-1}\})$;
- $u((n+1,\mu)) \coloneqq 1$ *for* $\mu \models \varphi_+$ *and* 0 *for* $\mu \not\models \varphi_+$;
- *For* $i \in +,-$, IS_i^k *is defined to be the* k*-th connected component of the equivalence relation*

$$\left\{\left((j,\mu),(j,\mu')\right) \in V^2 \mid \mathrm{P}(x_j) = i \wedge \mu \cap \mathrm{D}(j) = \mu' \cap \mathrm{D}(j)\right\};$$

- $A_i^k \coloneqq \{a_\top, a_\bot\}$, *with* $a_\top((j,\mu)) \coloneqq (j{+}1, \mu \cup \{x_{j+1}\})$ *and* $a_\bot((j,\mu)) \coloneqq (j{+}1,\mu)$ *for every* $(j,\mu) \in IS_i^k$.

By definition, CBGs define Boolean EFGs of chance. Moreover, since only nodes at the same depth can be in the same information set, it is clear that the EFGs thus defined are games with multi-agent perfect recall. it is also easy to see that a player $i \in \{+,-\}$ has perfect information if and only if $\mathrm{D}(x_j) = \{x_1,\ldots,x_{j-1}\}$ for all $x_j \in X$ such that $\mathrm{P}(x_j) = i$; this intuitively means player i observes all values chosen by Nature and the other player. It is also easy to see that a player $i \in \{+,-\}$ has perfect recall if and only if for all $x_j, x_{j'} \in X$, $j \leq j'$ and $\mathrm{P}(x_j) = \mathrm{P}(x_{j'}) = i$ implies $\mathrm{D}(x_j) \subseteq \mathrm{D}(x_{j'})$ (i.e. player i never forgets any value they have observed before) and $x_j \in \mathrm{D}(x'_j)$ (i.e. player i never forgets any value they have chosen before).

We will prove hardness results for CBGs with an arbitrary circuit φ_+. However, these hardness results even hold for CBGs the circuit φ_+ of which is restricted to certain languages (e.g. ROBDD, which stands for "reduced ordered binary decision diagram"; see Darwiche and Marquis (2002)), due to the following lemma, proven in Appendix B.1.3.

Lemma 4.3.3. *Let* $\gamma = \langle X, \mathrm{P}, \mathrm{D}, \varphi_+\rangle$ *be a CBG. Then there exists a CBG* $\gamma' \coloneqq \langle X', \mathrm{P}', \mathrm{D}', \varphi'_+\rangle$ *with* φ'_+ *in CNF (respectively DNF, ROBDD), which has the same maxmin value as* γ*, and such that (i)* γ' *is of no chance if and only if* γ *is of no chance; (ii) MAX and MIN have the same degree of imperfect information in* γ' *as in* γ*. Furthermore,* γ' *can be constructed from* γ *in polynomial time.*

Note that these restrictions are in some sense minimal: it is indeed easy to show that the computation of the maxmin value for a CBG is essentially trivial if the circuit φ_+ is restricted to be a single term (conjunction of literals) or a single clause (disjunction of literals).

Oracle Games

CBGs are intended to be a very constrained representation of games. At the other extreme, we now define oracle games, a minimally constrained representation for which we will show membership complexity results. Intuitively, an oracle game (respectively a valid oracle game) is a game with an exponential number of vertices $0, 1, \ldots, 2^n - 1$, represented in binary over n digits, for which all components are given by oracles (respectively by polynomial-space oracles that have a polynomial horizon).

Definition 4.3.4 (Oracle EFG)**.** *A two-player* oracle game *(abbreviated as OG) is a tuple of the form* $\mathtt{G} \coloneqq \langle n, \mathtt{C}, \mathtt{P}, \mathtt{p}, \mathtt{u}, \mathtt{IS}\rangle$*, where n is a positive integer, and* $\mathtt{C}, \mathtt{P}, \mathtt{p}, \mathtt{u}, \mathtt{IS}$ *are algorithms such that for all* $v, v' \in V(\mathtt{G}) \coloneqq \{0, 1, \ldots, 2^n - 1\}$*:*[10]

- *on input v,* $\mathtt{C}$ *returns an ordered list of elements of* $V(\mathtt{G})$ *(children of v);*
- *on input v,* $\mathtt{P}$ *returns one of* $0, +, -$ *(decision-maker at v);*
- *on inputs* v, v' *with* $\mathtt{P}(v) = 0$ *and* $v' \in \mathtt{C}(v)$*,* $\mathtt{p}$ *returns a rational number in* $[0, 1]$ *(probability of* v' *being the child of v drawn by Nature);*
- *on input v with* $\mathtt{C}(v) = \emptyset$*,* $\mathtt{u}$ *returns a rational number (utility of leaf v for MAX);*
- *on inputs* v, v' *and* $i \in \{+, -\}$*,* $\mathtt{IS}$ *returns a Boolean value (whether* v, v' *are in the same information set of i).*

Definition 4.3.5 (Valid OG)**.** *An OG* $\mathtt{G} \coloneqq \langle n, \mathtt{C}, \mathtt{P}, \mathtt{p}, \mathtt{u}, \mathtt{IS}\rangle$ *is said to be* valid *if the following conditions hold:*

1. *algorithms* $\mathtt{C}, \mathtt{P}, \mathtt{p}, \mathtt{u}, \mathtt{IS}$ *are all deterministic algorithms that run in space at most n and terminate in time at most* 2^n*;*
2. *the output of algorithms* $\mathtt{p}$ *and* $\mathtt{u}$ *is of size at most n (probabilities and utilities have a representation of linear size);*
3. *the binary relation* $\{(v, v') \in V(\mathtt{G})^2 \mid v' \in \mathtt{C}(v)\}$ *is a tree with the node* 0 *as the root;*
4. *for all sequences* $v_1 \in V(\mathtt{G}), v_2 \in \mathtt{C}(v_2), \ldots, v_k \in \mathtt{C}(v_{k-1})$*, k is at most n (that is, the game horizon is linear);*
5. *for all* $v \in V(\mathtt{G})$ *with* $\mathtt{P}(v) = 0$*,* $\sum_{v' \in \mathtt{C}(v)} \mathtt{p}(v, v') = 1$ *holds (*$\mathtt{p}$ *returns a probability distribution at chance nodes);*
6. *for* $i \in \{+, -\}$*,* $\mathtt{IS}(v, v', i) = 1$ *only if* $\mathtt{P}(v) = \mathtt{P}(v') = i$*;*
7. *for* $i \in \{+, -\}$*, the binary relation* $\{(v, v') \in V(\mathtt{G})^2 \mid \mathtt{IS}(v, v', i) = 1\}$ *is an equivalence relation such that for every* (v, v') *in this relation,* $|\mathtt{C}(v)| = |\mathtt{C}(v')|$ *holds (vertices in the same information set have the same number of children).*

Observe that we require the oracles to run in linear rather than polynomial space, and similarly we require a linear horizon. However, this assumption is without loss of generality up to polynomial-time reductions. Indeed, if an oracle runs in space n^c rather than n for some constant c, then one can define the OG game with n^c instead of n as its first component (akin to the idea of *padding* used in complexity theory). Similarly, up to a replacement of n by cn for some constant c, this definition is independent of the computational model on which the oracles are supposed to run.

The definition allows one to naturally capture families of games defined by the same rules (oracles) but different sizes (n); for instance, the family of Checkers games played on an $n \times n$ board. If such a family of OGs is valid, this means that it has "reasonable" game rules (essentially, computable in polynomial space). Accordingly, for complexity analyses, we define the *size* of a valid oracle game to be n; in particular, we do not count the representations of the oracles, for which we make no specific assumption.[11]

[10] If the number of nodes is not a power of 2, extra nodes are simply disconnected from the root 0.

[11] A reasonable encoding would be given by the input to a fixed Universal Turing Machine.

The interpretation of a valid OG as an EFG is straightforward. The actions at an information set can be denoted by integers in such a way that the k-th action maps every vertex v in the information set to the k-th child of v (which is well-defined due to Item 6 and 7 of Definition 4.3.5). Moreover, given the requirements on the oracle, the following result is straightforward.

Lemma 4.3.6. *The EFG of chance defined by a given valid OG* $\mathtt{G}$ *can be computed in deterministic exponential time. In particular, this EFG has at most exponential size in the size n of* $\mathtt{G}$.

Let us emphasise that it can be decided in polynomial space whether a given OG is valid, by verifying that no property is violated; for instance, it can be checked that $\mathtt{p}$ runs in space at most n by enumerating all pairs of vertices and for each one, running the algorithm until more than n space, or more than 2^n time, is used (if ever); all of this can be done in polynomial space.

Similarly, it can also be decided in polynomial space whether a player in a given OG has perfect information, perfect recall, or multiagent perfect recall, by verifying that there is no counterexample: for PI, two nodes in the same information set; for PR, an invalid pair of paths to the same information set; for MA-PR, a path intersecting the same information set twice.

Relationships between compact representations

It is easy to see that given a CBG $\gamma \coloneqq \langle X, \mathrm{P}, \mathrm{D}, \varphi_+ \rangle$ with n_γ variables, an OG $\mathtt{G} \coloneqq \langle n_{\mathtt{G}}, \mathtt{C}, \mathtt{P}, \mathtt{p}, \mathtt{u}, \mathtt{IS} \rangle$ that defines the same EFG as γ can be computed efficiently.

Indeed, the vertices in the EFG can be encoded efficiently over $n_\gamma + \log n_\gamma$ bits, by encoding the level i of a vertex over $\log n_\gamma$ bits, and the assignment of the previous variables over n_γ bits. Given this propositional representation, the oracles in $\mathtt{G}$ can be defined to be the straightforward polynomial-space algorithms that, given input in this representation, retrieve the corresponding information from γ; for instance, $\mathtt{u}$ can be implemented by evaluating φ_+ on the values of all variables stored in the representation of the node. Taking $cn_\gamma^{c'}$ to be the maximum space used by these algorithms on the computational model at hand and defining $n_{\mathtt{G}} \coloneqq \max(n_\gamma + \log n_\gamma, cn_\gamma^{c'})$ indeed yields a valid OG (observe that the horizon of the underlying EFG is n_γ).

To summarise, a family of CBGs can be transformed in polynomial time into a family of valid OGs that define the same family of EFGs. We will in particular use this result to show that if computing maxmin is hard for CBGs of some type, then it is hard as well for valid OGs of this type.

In addition, since CBG is a relatively minimal formalism to represent compact games, while OG is a relatively powerful one, if we prove that computing the maxmin value for CBGs is as hard as for OGs, then it implies that for every other intermediate formalism, the complexity of computing the maxmin value is as hard as for both CBG and OG.

More concretely, for membership results, we prove them for OG; for hardness results, we prove them for CBG. We leave to future work the precise relationship between oracle games as we define them, and standard compact languages for defining games, such as the Game Description Language GDL (Genesereth and Thielscher, 2014). We conjecture that propositional GDL (restricted to games with a polynomial horizon) can be reduced to OG, while full GDL cannot (even with the restriction to a polynomial horizon) due to the computational complexity of generating states (nodes

in the tree) using logic programming. However, the precise connection remains open; a good starting point may be the complexity results by Bonnet and Saffidine (2014).

4.3.2 Summary of results

In the remainder of this section, we consider the following variant of Pure Maxmin.

Definition 4.3.7 (Pure C-Maxmin)**.** *Let $\mathcal{G}$ be a class of zero-sum EFGs. Then* Pure C-Maxmin($\mathcal{G}$) *is the following decision problem.*

Input *A valid oracle game* $\mathtt{G}$ *that defines an EFG $G \in \mathcal{G}$, and a rational number m.*

Output *Decide whether $\underline{v}_+(\Sigma^{\mathrm{P}}_+, \Sigma^{\mathrm{P}}_-) \geq m$ holds in G.*

Recall that the size of a valid OG $\mathtt{G} := \langle n, \mathtt{C}, \mathtt{P}, \mathtt{p}, \mathtt{u}, \mathtt{IS}\rangle$ is defined to be n. The complexity of Pure C-Maxmin($\mathcal{G}$) is summarised in Table 4.2; see Subsection 4.2.1 for hints about how to read it. Observe that this table is parallel to Table 4.1, in the sense that all polynomial-time (respectively NP-complete, Σ^{P}_2-complete) problems become PSPACE-complete (respectively NEXP-complete, $\mathsf{NEXP}^{\mathsf{NP}}$-complete) under compact representations. However, except for membership results, this is not per definition of succinct representations, and we indeed have to prove all hardness results (even if the proofs are simple).

	No chance	Chance		
MAX \ MIN	PI/PR/MA-PR	PI	PR	MA-PR
PI	PSPACE-c [h: 4.3.12]	PSPACE-c [m: 4.3.8]	**NEXP-c [h: 4.3.14]**	**$\mathsf{NEXP}^{\mathsf{NP}}$-c [h: 4.3.16]**
PR	**PSPACE-c [m: 4.3.9]**	**NEXP-c [h: 4.3.15]**	**NEXP-c**	**$\mathsf{NEXP}^{\mathsf{NP}}$-c**
MA-PR	NEXP-c [h: 4.3.13]	NEXP-c	NEXP-c [m: 4.3.10]	**$\mathsf{NEXP}^{\mathsf{NP}}$-c** [m: 4.3.11]

Table 4.2: Complexity of Pure C-Maxmin. All hardness results hold even under the restriction to CBGs with CNF (respectively DNF, ROBDD) circuits.

4.3.3 Membership results

Proposition 4.3.8. Pure C-Maxmin *is in* PSPACE *for valid OGs of chance in which both MAX and MIN have PI.*

Proof. In this case, we can use backward induction to compute the maxmin value (Proposition 4.2.9). Since the game has a polynomial horizon by assumption, and the children of a node can be iterated over in PSPACE, this can indeed be done in PSPACE. □

Proposition 4.3.9. Pure C-Maxmin *is in* PSPACE *for valid OGs of no chance in which MAX has PR and MIN has MA-PR.*

Proof. We consider the algorithm of Proposition 4.2.4, which computes the maxmin value by a bottom-up induction on the game tree. Since the game tree of an oracle game has polynomial depth and the children of each information set can be iterated over in PSPACE, this computation can be done in PSPACE from the oracle representation. □

The following two results are obtained directly by using Lemma 4.3.6 to first compute, in deterministic exponential time, the EFG defined by the given valid OG, then running a nondeterministic polynomial-time algorithm (Proposition 4.2.3), respectively a nondeterministic polynomial-time algorithm with an NP-oracle (Proposition 4.2.14) on this EFG of exponential size.

Proposition 4.3.10. Pure C-Maxmin *is in* NEXP *for valid OGs of no chance in which MAX and MIN have MA-PR, and for valid OGs of chance in which MAX has MA-PR and MIN has PR.*

Proposition 4.3.11. Pure C-Maxmin *is in* $\mathsf{NEXP}^{\mathsf{NP}}$ *for valid OGs of chance in which both MAX and MIN have MA-PR.*

4.3.4 Hardness results

As discussed above, we give all hardness results for compact Boolean games. Since all proofs use a polynomial-time reduction from a decision problem to Pure C-Maxmin for a family of CBGs, all these results also imply that Pure C-Maxmin is at least as hard for the corresponding family of valid OGs, as discussed in Section 4.3.1.

Proposition 4.3.12. Pure C-Maxmin *is* PSPACE*-hard for CBGs of no chance in which both MAX and MIN have PI.*

Proof. It is easy to see that given a QBF $\mathrm{Q}_1 x_1 \, \mathrm{Q}_2 x_2 \cdots \mathrm{Q}_n x_n \, \varphi(x_1, x_2, \ldots, x_n)$, the CBG $\langle X, \mathrm{P}, \mathrm{D}, \varphi \rangle$, with $X = (x_1, \ldots, x_n)$, $\mathrm{P}(i) = +$ for $\mathrm{Q}_i = \exists$ and $\mathrm{P}_i = -$ for $\mathrm{Q}_i = \forall$, and $\mathrm{D}(x_i) = \{x_1, \ldots, x_{i-1}\}$ for all i, is a game of no chance and with perfect information. Moreover, it has a maxmin value of 1 if and only if the QBF is valid. We conclude using the PSPACE-hardness of deciding the validity of a QBF (Stockmeyer and Meyer, 1973). □

Proposition 4.3.13. Pure C-Maxmin *is* NEXP*-hard for CBGs of no chance in which MAX has MA-PR and MIN has PR.*

Proof. Peterson et al. (2001, Theorem 5.2.1) show that deciding the validity of a DQBF of the form $\forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2) \, \varphi(x_1, y_1, x_2, y_2)$ is already NEXP-hard. By defining $X \coloneqq (x_1, x_2, y_1, y_2)$, $\mathrm{P}(x_1) = \mathrm{P}(x_2) \coloneqq -$, $\mathrm{P}(y_1) = \mathrm{P}(y_2) \coloneqq +$, and $\mathrm{D}(x_1) = \mathrm{D}(x_2) \coloneqq \emptyset$, $\mathrm{D}(y_1) \coloneqq \{x_1\}$, $\mathrm{D}(y_2) \coloneqq \{x_2\}$, we get the CBG $\langle X, \mathrm{P}, \mathrm{D}, \varphi \rangle$, for which it is easy to see that it has a maxmin value of 1 if and only if the DQBF is valid, which concludes. □

Proposition 4.3.14. Pure C-Maxmin *is* NEXP*-hard for CBGs of chance in which MAX has PI and MIN has PR.*

Proof. We adapt the proof of Proposition 4.2.10 to CBGs, by giving a reduction from Succinct 4-Colourability.[12] An instance of this problem is a circuit ϕ with $2n$ inputs $x_1, \ldots, x_n, y_1, \ldots, y_n$ encoding integers x, y; ϕ describes the graph G with 2^n vertices $0, \ldots, 2^n - 1$, which has an edge (x, y) if and only if the circuit outputs 1 on input (x, y) or (y, x) (or both), and ϕ is a positive instance if and only if G is 4-colourable. This problem is NEXP-complete (Balcázar et al., 1992, Corollary 6).[13]

[12] We choose 4 colours for simplicity, as 4 is a power of 2. It is clear that the proof of Proposition 4.2.10 goes through with 4-colourability instead of 3-colourability.

[13] In general, the number of vertices in the graph is in $\{2^{n-1}, 2^{n-1} + 1, \ldots, 2^n - 1\}$, but a graph can always be padded with isolated vertices to ensure that the number of vertices is a power of 2, and this does not change 4-colourability; accordingly, the succinct representation ϕ can be modified in polynomial time to represent this, by replacing ϕ with $\phi^{x,y} \wedge \phi$, where $\phi^{x,y}$ tests that x and y are less than the initial number of vertices.

Given an instance of the succinct version, we define the ordered list of $3n+4$ Boolean variables $(x_1^v, \dots, x_n^v, x_1^{c_+}, x_2^{c_+}, x_1^e, \dots, x_n^e, x_{n+1}^e, \dots, x_{2n}^e, x_1^{c_-}, x_2^{c_-})$ meant to encode in binary the vertex v chosen by Nature, the colour c_+ chosen by MAX, the two vertices of the edge e chosen by MIN, and the colour c_- chosen by MIN, respectively.

We moreover define

$$\varphi_+ := \vee \begin{cases} \left(\neg\phi(x_1^e, \dots, x_n^e, x_{n+1}^e, \dots, x_{2n}^e) \wedge \neg\phi(x_{n+1}^e, \dots, x_{2n}^e, x_1^e, \dots, x_n^e)\right) \\ \neg\left(\left(\bigwedge_{j=1}^n (x_j^v \leftrightarrow x_j^e) \vee \bigwedge_{j=1}^n (x_j^v \leftrightarrow x_{n+j}^e)\right) \wedge \bigwedge_{j=1}^2 (x_j^{c_+} = x_j^{c_-})\right) \end{cases}$$

where the first line encodes that MAX wins if MIN chooses a non-edge, and the second one encodes the winning conditions of the proof of Proposition 4.2.10.

Finally, we define players and dependencies as follows; for all j:

- $\mathrm{P}(x_j^v) := 0$, $\mathrm{P}(x_j^{c_+}) := +$, $\mathrm{P}(x_j^e) := -$, $\mathrm{P}(x_j^{c_-}) := -$;
- $\mathrm{D}(x_j^v) = \emptyset$;
- $\mathrm{D}(x_1^{c_+}) = \{x_1^v, \dots, x_n^v\}$, $\mathrm{D}(x_2^{c_+}) = \mathrm{D}(x_1^{c_+}) \cup \{x_1^{c_+}\}$;
- $\mathrm{D}(x_j^e) = \{x_1^e, \dots, x_{j-1}^e\}$;
- $\mathrm{D}(x_1^{c_-}) = \{x_1^e, \dots, x_{2n}^e\}$, $\mathrm{D}(x_2^{c_-}) = \mathrm{D}(x_1^{c_-}) \cup \{x_1^{c_-}\}$.

It is easy to see that if the circuit ϕ describes a graph G, then the compact Boolean game $\langle X, \mathrm{P}, \mathrm{D}, \varphi_+\rangle$ encodes the same game as the explicit game built from G in the proof of Proposition 4.2.10, with the exception that MIN can choose a non-edge $e \notin E$; however, since MAX obtains the maximal payoff 1 (first line of φ_+) in this case, MIN never has a strict incentive to choose a non-edge and hence, this does not change the maxmin value of the game. Hence, we have indeed given a polynomial reduction from Succinct 4-Colourability to Pure C-Maxmin for CBGs of chance in which MAX has PI and MIN has PR. □

By using the same proof as for Proposition 4.3.14, but having MIN's variables ordered first and resorting to Proposition 4.2.11 instead of Proposition 4.2.10, we obtain the same result for the dual setting.

Proposition 4.3.15. Pure C-Maxmin *is* NEXP*-hard for CBGs of chance in which MAX has PR and MIN has PI.*

Finally, we turn to the case in which MIN only has multi-agent perfect recall.

Proposition 4.3.16. Pure C-Maxmin *is* $\mathsf{NEXP}^{\mathsf{NP}}$*-hard for CBGs of chance in which MAX has PI and MIN has MA-PR.*

Proof. The proof is similar to the proof of Proposition 4.2.12, but we consider the version of Tiling Extension in which m is encoded in binary, which is known to be $\mathsf{NEXP}^{\mathsf{NP}}$-hard (Goldsmith and Mundhenk, 2007, Theorem 2.2). We build the same game as in Proposition 4.2.12, but as a compact Boolean game, as follows.

We write $p := \lceil \log m \rceil$ and $q := \lceil \log |D| \rceil$. The game has variables $x_1^c, \dots, x_p^c$, encoding the bits of a column number c, $x_1^d, \dots, x_q^d$, encoding the choice of a tile $d \in D$, and similarly $x_1^{r_1}, \dots, x_p^{r_1}, x_1^{c_1}, \dots, x_p^{c_1}, x_1^{r_2}, \dots, x_p^{r_2}, x_1^{c_2}, \dots, x_p^{c_2}$ encoding the choice of positions (r_1, c_1) and (r_2, c_2), and $x_1^{d_1}, \dots, x_q^{d_1}, x_1^{d_2}, \dots, x_q^{d_2}$, encoding the choice of tiles $d_1, d_2 \in D$. These variables are ordered as they are listed. At the end

of this ordering, a polynomial number of variables $x_1^u, \ldots, x_n^u$ are added, as required by the transformation of the game to a Boolean game.

The players and dependencies are as follows, in one-to-one correspondence with the description of the game tree in the proof of Proposition 4.2.12:

1. for $j = 1, \ldots, p$, $\mathrm{P}(x_j^c) := 0$ (and hence, $\mathrm{D}(x_j^c) := \emptyset$);
2. for $j = 1, \ldots, q$, $\mathrm{P}(x_j^d) := +$ and $\mathrm{D}(x_j^d) := \{x_1^c, \ldots, x_p^c\} \cup \{x_1^d, \ldots, x_{j-1}^d\}$;
3. for $t \in \{r_1, c_1, r_2, c_2\}$ and $j = 1, \ldots, p$, $\mathrm{P}(x_j^t) := 0$;
4. for $i = 1, 2$ and $j = 1, \ldots, q$, $\mathrm{P}(x_j^{d_i}) := -$ and $\mathrm{D}(x_j^{d_i}) := \{x_1^{r_i}, \ldots, x_p^{r_i}\} \cup \{x_1^{c_i}, \ldots, x_p^{c_i}\} \cup \{x_1^{d_i}, \ldots, x_{j-1}^{d_i}\}$.

We finally define φ_+. First observe from the proof of Lemma B.1.5 that if utility u is assigned to a set of leaves satisfying φ^u in the original game, then there is a polynomial-size disjunction φ_b^u of terms over the variables $x_1^u, \ldots, x_m^u$ such that 1 (respectively 0) is assigned to the set of leaves satisfying $\varphi^u \wedge \varphi_b^u$ (respectively $\varphi^u \wedge \neg\varphi_b^u$) in the corresponding Boolean game.[14]

Now define the following propositional formulae, for $\vec{x} = (x_1, \ldots, x_k)$ and $\vec{y} = (y_1, \ldots, y_k)$:

- $\varphi^1(\vec{x}) := \bigwedge_{j=1}^{k-1} \neg x_j \wedge x_k$, satisfied if and only if $\vec{x}$ encodes the integer 1;
- $\varphi^=(\vec{x}, \vec{y}) := \bigwedge_{j=1}^{k} (x_j \leftrightarrow y_j)$, satisfied if and only if $\vec{x}$ and $\vec{y}$ are the binary encodings of the same integer;
- $\varphi^{+1}(\vec{x}, \vec{y}) := \bigvee_{j=1}^{k} \left(\bigwedge_{j'=1}^{j-1} (x_{j'} \leftrightarrow y_{j'}) \wedge x_j \wedge \neg y_j \wedge \bigwedge_{j'=j+1}^{k} (\neg x_{j'} \wedge y_{j'}) \right)$, satisfied if and only if $\vec{x}$ is the encoding of the integer encoded by $\vec{y}$, plus 1;
- $\varphi^{[m]}(\vec{x})$ (respectively $\varphi^D(\vec{x})$) for formulae satisfied if and only if $\vec{x}$ encodes an integer in $\{1, \ldots, m\}$ (respectively in $\{1, \ldots, |D|\}$);[15]
- $\varphi^H(\vec{x}, \vec{y})$ (respectively $\varphi^V(\vec{x}, \vec{y})$) for a formula satisfied if and only if the tiles d_x, d_y encoded by $\vec{x}, \vec{y}$ are in the H (respectively V) relation; such polynomial size formulae can be obtained as the disjunction of the description of all pairs $\vec{x}, \vec{y}$ (as terms), since H and V are assumed to be explicitly given in the input to Succinct Tiling Extension.

With these formulae in hand, we define the following formulae, in one-to-one correspondence with the description of the utilities in Proposition 4.2.12:[16]

- $\psi_1 := \varphi_b^{2m^5} \wedge r_1 = r_2 \wedge c_1 = c_2 \wedge \neg(d_1 = d_2)$,
- $\psi_2 := \varphi_b^{2m^4} \wedge \neg\psi_1 \wedge \bigvee_{i=1,2} \left(r_i = 1 \wedge c_i = c \wedge \neg(d_i = d)\right)$,
- $\psi_3 := \varphi_b^{-m^4} \wedge \neg\psi_1 \wedge \neg\psi_2 \wedge r_i = 1 \wedge r_2 = 1 \wedge (c_2 = c_1 + 1) \wedge \neg H(d_1, d_2)$,

[14] For instance, for the reformulation of Figure B.1 (page 177), we would have $\varphi_b^{5/8} = x_1^{5/8} \vee (\neg x_1^{5/8} \wedge \neg x_2^{5/8} \wedge x_3^{5/8})$, encoding the two paths to a 1-leaf in the tree.

[15] For instance, for $m = 5$, which has binary encoding 101, $\varphi^{[m]}$ is $\neg x_1 \vee (x_1 \wedge (\neg x_2 \wedge (\neg x_3 \vee x_3))) \equiv \neg x_1 \vee \neg x_2$.

[16] To facilitate reading, we use shorthand notation, e.g. $r_1 = r_2$ for $\varphi^=(\vec{x}^{r_1}, \vec{x}^{r_2})$, but keeping in mind that these are indeed polynomial-size formulae.

- $\psi_4 := \varphi_b^1 \wedge \neg\psi_1 \wedge \neg\psi_2 \wedge \neg\psi_3 \wedge r_2 = r_1 \wedge \neg(r_1 = 1) \wedge (c_2 = c_1 + 1) \wedge \neg H(d_1, d_2)$,
- $\psi_5 := \varphi_b^1 \wedge \neg\psi_1 \wedge \neg\psi_2 \wedge \neg\psi_3 \wedge \neg\psi_4 \wedge (r_2 = r_1 + 1) \wedge (c_2 = c_1) \wedge \neg V(d_1, d_2)$,
- $\psi_6 := \varphi_b^0 \wedge \neg\psi_1 \wedge \neg\psi_2 \wedge \neg\psi_3 \wedge \neg\psi_4 \wedge \neg\psi_5$.[17]

Finally, φ_+ is defined to be

$$\bigvee_{i=1,2} \neg\varphi^D(\vec{x}^{d_i}) \vee \left(\varphi^{[m]}(\vec{x}^c) \wedge \varphi^D(\vec{x}^d) \wedge \bigwedge_{i=1,2} (\varphi^{[m]}(\vec{r}_i) \wedge \varphi^{[m]}(\vec{c}_i)) \wedge \bigwedge_{k=1}^{6} \psi_k \right).$$

It is easy to see that φ_+ encodes the same rewards as the game built in the proof of Proposition 4.2.12, except in the following cases:

- when MIN chooses non-existent tiles, but since this ensures the maximal payoff of 1 for MAX (via the first disjunct of φ_+), MIN never has a strict incentive to do so, hence this does not change the maxmin value of the game,
- otherwise, when MAX chooses a non-existent tile, but since this gives them the minimal payoff of 0 (the second conjunct inside the parentheses), again it never has a strict incentive to do so,
- otherwise, when Nature chooses non-existent row or columns, which yields a neutral payoff of 0; this happens with probability $p := 1 - \frac{m^5}{2^{5\lceil \log m \rceil}}$, independently of MIN's and MAX' strategies.

From this, it follows that the game of Proposition 4.2.12 has a maxmin value of k if and only if the game above has a maxmin value of $k(1 - p)$, hence we indeed have a reduction from Succinct Tiling Extension to Pure C-Maxmin. □

4.4 Complexity against opponent models

We now consider the situation in which MIN is only allowed to choose from a finite set of behaviour strategies. This setting corresponds to playing a game against an opponent whose behaviour or reasoning is captured by a model we know. Such a model is called an *opponent model* (cf. Section 2.5). The same setting also captures the problem of planning in uncertain environments or with adversarial cost functions (McMahan et al., 2003).

Concretely, we study the following variant of Pure Maxmin. As in Section 4.2, we consider the complexity for EFGs.

Definition 4.4.1 (Pure OM-Maxmin)**.** *Let $\mathcal{G}$ be a class of zero-sum EFGs. Then* Pure OM-Maxmin($\mathcal{G}$) *is the following decision problem.*

Input *An EFG $G \in \mathcal{G}$, a rational number m, and a finite set $\Sigma_-^{\mathrm{O}} \subseteq \Sigma_-^{\mathrm{B}}$ of MIN's behaviour strategies in G.*

Output *Decide whether $\underline{v}_+(\Sigma_+^{\mathrm{P}}, \Sigma_-^{\mathrm{O}}) \geq m$ holds in G.*

[17] Lemma B.1.5 applied to this example will rescale the 0-reward to another value, so we must indeed take the 0-reward into account here.

We define the size of the input of this problem to be the sum of $\|G\|$, $\|m\|$, and the sizes of the strategies in Σ_{-}^{O}; the size of a behaviour strategy $\pi_{-} \in \Sigma_{-}^{O}$ is defined to be the sum of the sizes of the rational numbers (probabilities) that define π_{-}, over all MIN's nodes in the game tree.

In the following, strategies in $\Sigma_{-}^{O} \subseteq \Sigma_{-}^{B}$ will be referred to as opponent models (abbreviated as OMs). A game with OMs is said to be of no chance, if the original game is of no chance, and all OM strategies in Σ_{-}^{O} are pure strategies. This terminology is consistent, since a player with a non-pure OM strategy is indistinguishable from Nature, a player whose behaviour strategy is known.

4.4.1 Summary of results

The complexity of PURE OM-MAXMIN($\mathcal{G}$) is summarised in Table 4.3; see Subsection 4.2.1 for hints about how to read it. For each degree of imperfect information for MAX (PI/PR/MA-PR), the first row corresponds to games of no chance and the second one to games of chance. Note that it no longer makes sense to distinguish between different degrees of imperfect information of MIN, but it makes sense to study the complexity with respect to the number of OMs (1, a constant but at least 2, or unbounded).

Chance \ $\lvert\Sigma_{-}^{O}\rvert$ / MAX	1	constant (≥ 2)	unbounded
PI	P	P	P
	P	**NP-c [h: 4.4.8]**	**NP-c**
PR	P	P	**P [m: 4.4.5]**
	P [m: 4.4.3]	**NP-c**	NP-c
MA-PR	P [m: 4.4.3]	**P [m: 4.4.6]**	NP-c [h: 4.4.7]
	NP-c [h: 4.4.4]	NP-c	NP-c [m: 4.4.2]

Table 4.3: Complexity of PURE OM-MAXMIN. All hardness results hold even under the restriction to Boolean games with at most 2 agents for MAX.

Since we only consider pure strategies for MAX, this problem is trivially in NP.

Proposition 4.4.2. PURE OM-MAXMIN *is in* NP.

Proof. One can guess a pure strategy of MAX, then verify that it yields an expected payoff no less than the threshold against all the OMs given in the input, all in time linear in the size of the input. □

This provides an upper bound for all cases in Table 4.3. Other results will be shown in the following parts.

4.4.2 Complexity of best responses in EFGs

We begin by considering the case $|\Sigma_{-}^{O}| = 1$, i.e. MIN's behaviour strategy is fixed (or, equivalently, MAX knows MIN's probability of choosing each strategy from a finite set of OMs). Under this circumstance, a two-player game is transformed into a one-player game, which has no chance if the original game has no chance and MIN's known strategy is a pure strategy.

We denote the only strategy in Σ_-^{O} by π_-. Clearly, the maxmin value $\underline{v}_+(\Sigma_+^{\mathrm{P}}, \Sigma_-^{\mathrm{O}}) = \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \mathcal{U}_+(s_+, \pi_-)$ is the value of MAX's pure best responses to π_-. Hence, studying the complexity of the case $|\Sigma_-^{\mathrm{O}}| = 1$ is equivalent to studying the complexity of finding the best responses of a player.

The best responses are easy to compute when MAX has perfect recall or there is no chance.

Proposition 4.4.3. *(Koller and Megiddo, 1992, Section 3.3)* Pure OM-Maxmin *with only one OM is decidable in linear time, and is* a fortiori *in* P*, for EFGs of no chance in which MAX has MA-PR, and for EFGs of chance in which MAX has PR.*

However, if MAX only has MA-PR and there is chance (due to chance nodes and/or due to MIN's behaviour strategy π_-), then the problem becomes intractable.

Proposition 4.4.4. Pure OM-Maxmin *with only one OM is* NP*-hard for EFGs of chance in which MAX has MA-PR. The result holds even under the restriction to* 2 *agents for MAX and to Boolean games.*

Proof. Consider the same reduction from 3-Colouring as in the proof of Proposition 4.2.8. In the game obtained by the reduction, consider MIN's uniformly random strategy π_- (i.e. MIN picks any pair of $(v_1, v_2) \in V^2$ with a probability of $1/|V|^2$).

One can easily verify that if (V, E) has a legal 3-colouring, then MAX can play a strategy corresponding to a legal 3-colouring to ensure a payoff of 1 against MIN's strategy π_-. Conversely, if the graph is not 3-colourable, then for every pure strategy of MAX, the expected payoff against π_- is strictly less than 1, since any inconsistency between the two agents of MAX or any monochromatic edge in MAX's chosen colouring will be detected by MIN's strategy π_- with a non-zero probability, in which case MAX gets 0 as payoff. Hence, the game has a maxmin value of at least 1 if and only if the graph is 3-colourable. □

4.4.3 Complexity of EFGs of no chance with multiple OMs

We now consider the case $|\Sigma_-^{\mathrm{O}}| \geq 2$, starting with games of no chance. In this problem setting, the OMs are pure strategies of MIN. Hence, we can write $\Sigma_-^{\mathrm{O}} = \{s_{-,1}, \ldots, s_{-,|\Sigma^{\mathrm{O}}|}\}$, where $s_{-,i} \in \Sigma_-^{\mathrm{P}}$ for $1 \leq i \leq |\Sigma_-^{\mathrm{O}}|$.

Proposition 4.4.5. Pure OM-Maxmin *with multiple OMs is in* P *for EFGs of no chance in which MAX has PR.*

Proof. Let $n = |\Sigma_-^{\mathrm{O}}|$. We propose the procedure depicted in Algorithm 1, in which we write $a(v_q)$ for the child of v_q reached by taking the action a.

The Procedure `Maxmin` takes as input $\{(s_{-,i}, v_i) \mid i = 1, \ldots, n\}$, a multi-set of vertices, each associated with an OM of MIN. It returns true if there is a pure strategy for MAX with maxmin value at least m in the subgame induced by these vertices and OMs.

For this, Procedure `Maxmin` first computes the next decision nodes of MAX reached by simulating each OM $s_{-,i}$ starting from the associated vertex v_i. If along the way a leaf with value less than m is encountered, then it returns false, since avoiding this leaf is out of control of MAX. If a leaf with value at least m is encountered, then MAX has nothing to do against this OM; hence this leaf can be ignored (which is done tacitly in Algorithm 1). Otherwise, an internal node v_i' of MAX is encountered, from which MAX must make a decision. Hence, we gather each such nodes and partition them

Algorithm 1: Polynomial-time algorithm for PURE OM-MAXMIN against n OMs for EFGs of no chance in which MAX has PR.

```
return Maxmin({(s_{-,i}, r) | i = 1, ..., n})

def Maxmin({(s_{-,i}, v_i) | i ∈ I}):
    V' ← ∅
    for i ∈ I:
        v'_i ← v_i
        while v'_i is not a leaf and P(v'_i) = −:
            v'_i ← s_{-,i}(v'_i)
        if v'_i is a leaf and u_+(v'_i) < m:
            return False
        elif v' is not a leaf:
            V' ← V' ∪ {(s_{-,i}, v'_i)}
    V'_1, ..., V'_p ← partition of V' by the information sets of MAX
    for j = 1, ..., p:
        action_found ← False
        for each MAX's action a available at the information set containing V'_j:
            V''_j ← {(s_{-,q}, a(v_q)) | (s_{-,q}, v_q) ∈ V'_j}
            if Maxmin(V''_j) returns True:
                action_found ← True
                break
        if action_found = False:
            return False
    return True
```

according to the information set of MAX containing them, yielding $V'_1, \ldots, V'_p$; for each one, we try to find an action for MAX from which there is a strategy with maxmin value at least m, using a recursive call.

To show that this procedure is correct, assume first that MAX has a strategy with maxmin value at least m. Then, by definition, this strategy takes the same action at all nodes within the same information set, and against every OM the unique leaf reached has value at least m. Hence, Algorithm 1 indeed returns true.

Conversely, assume that the algorithm returns true. Then there is a traversal of the game tree such that at each decision node of MAX an action is taken, in such a way that against each OM, a leaf with payoff at least m is reached. Now these actions together indeed form a pure strategy for MAX since (i) by construction, whenever two nodes in the same information set are encountered at the same level of recursion, the same action is taken, and (ii) it cannot be the case that two successful recursive calls at the same level encounter the same information set of MAX; indeed, otherwise there would be two different decisions for MAX (at V'_{j_1} and V'_{j_2}) leading to the same information set, violating the assumption of perfect recall for MAX.[18] It follows that the actions in independent recursive calls can indeed be chosen independently of each other. □

If MAX only has multi-agent perfect recall, Algorithm 1 no longer works. However,

[18]This can also be shown by noticing that MAX's information sets form a forest due to their perfect recall (Koller and Megiddo, 1992, Proposition 3.1).

if the number of OMs is bounded by a constant, then the decision problem is still in P.

Proposition 4.4.6. *For every $k \geq 1$,* Pure OM-Maxmin *with k OMs is in* P *for EFGs of no chance in which MAX has MA-PR.*

Proof. Let us write $\Sigma_-^{\mathrm{O}} = \{s_{-,1}, \ldots, s_{-,k}\}$, where $s_{-,i} \in \Sigma_-^{\mathrm{P}}$ for $1 \leq i \leq k$. We first observe that for a fixed pure strategy of MAX and a fixed OM, a unique leaf of the game tree T is reached. Hence, the outcome of a given strategy s_+ of MAX against all OMs can be represented by a k-tuple of leaves $(l_1, \ldots, l_k)$ (possibly with repetitions) such that l_i is the leaf reached by the playout under the profile $(s_+, s_{-,i})$. For every constant k, there are only $O(|\mathrm{L}(T)|^k)$ such tuples. Hence, we can enumerate them to decide whether there is one k-tuple for which (i) the value $\min_{1 \leq i \leq k} u_+(l_i)$ is at least m and (ii) the tuple indeed corresponds to the outcomes of some pure strategy of MAX against the k OMs.

Algorithm 2: Polynomial-time algorithm for verifying that a k-tuple $(l_1, \ldots, l_k)$ is reachable under some pure strategy of MAX against the k OMs.

```
s_+ ← empty mapping from 𝓘𝓢_+ to actions
for i = 1, ..., k:
    pl ← the unique path from r to l_i
    for v ∈ pl:
        a ← the action taken at v in pl
        if P(v) = −:
            if a ≠ s_{−,i}(v):
                return False      /* l_i is not consistent with s_{−,i} */
            continue          /* v is good, look at the next one */
        IS_+ ← the information set of MAX containing v
        if s_+(IS_+) is not set yet:
            s_+(IS_+) ← a
        elif s_+(IS_+) ≠ pl(v):
            return False              /* no s_+ can reach (l_1, ..., l_k) */
return True
```

It remains to show that (ii) can be decided in polynomial time. The whole algorithm is given as Algorithm 2. We first check that on the unique path from the root to l_i, all decisions taken at MIN's nodes are indeed as prescribed by the i-th OM $s_{-,i}$. We then check that these k paths indeed prescribe the same action at every information set of MAX. Clearly, Algorithm 2 runs in polynomial time (more precisely, in $O(k|T|)$ time), which completes the proof. □

It turns out that a constant number of OMs is essentially the best we can do when MAX only has multi-agent perfect recall.

Proposition 4.4.7. Pure OM-Maxmin *with multiple OMs is* NP*-hard for EFGs of no chance in which MAX has MA-PR. The results hold even under the restriction to* 2 *agents for MAX, to Boolean games, and to polynomially (in the size of game tree) many OMs.*

Proof. Consider again the reduction in Proposition 4.2.8. Notice that in the game obtained by the reduction, MIN has $|V|^2$ pure strategies, one for each pair of vertices.

If we take all these pure strategies as OMs, then a graph has a legal 3-colouring if and only if MAX has a pure strategy with payoff 1 against all these OMs. □

Another way to interpret this result is by comparing it to Proposition 4.4.4: the proof there is essentially the same, but using a single behaviour OM, with a stochastic behaviour corresponding to a uniform mixture of the pure OMs used in the proof of Proposition 4.4.7.

4.4.4 Complexity of EFGs of chance with multiple OMs

We now turn to the case of multiple OMs, for games of chance. Surprisingly, in the presence of chance, computing the pure maxmin value against only 2 OMs is already NP-hard, even if MAX has perfect information.

Proposition 4.4.8. Pure OM-Maxmin *with* 2 *OMs is* NP*-hard for EFGs of chance in which MAX has PI. The results hold even under the restriction to Boolean games.*

Proof. We give a reduction from the NP-complete problem Subset Sum, which is defined as follows:

Input A multi-set of natural numbers $S = \{i_1, \ldots, i_n\}$, a natural number k.

Output Decide whether there exists a subset $J \subseteq S$ that sums up to k ($\sum_{j \in J} j = k$).

Let $S = \{i_1, \ldots, i_n\}$ and k form an instance of Subset Sum. We build a game in which, intuitively, a strategy of MAX is a subset J of S, one OM verifies $\sum_{j \in J} j \geq k$, and the other verifies $\sum_{j \in J} j \leq k$.

Concretely, consider the following game:

Players Nature; MAX with perfect information; MIN with two OMs, $\pi_{-,\leq}$ and $\pi_{-,\geq}$.

Game tree See Figure 4.1. At the root, Nature chooses uniformly at random an element $j \in S$. MAX then chooses between ✓ (encoding the choice of some $J \ni j$) or × ($J \not\ni j$). Finally, MIN chooses ≤ or ≥: MIN always chooses ≤ under the OM $\pi_{-,\leq}$, and always chooses ≥ under $\pi_{-,\geq}$.

Payoffs for MAX For each Nature's choice $j \in S$, MAX's payoff is as follows:

- If MIN has chosen ≥, MAX receives nj if they have chosen ✓, otherwise 0.
- If MIN has chosen ≤, MAX receives $2k - nj$ if they have chosen ✓, otherwise $2k$.

Maxmin The threshold of maxmin value is k.

The construction is polynomial-time in the input (S, k). Indeed, the game tree is of size $O(|S|)$. In addition, the construction yields an EFG of chance with 2 OMs in which MAX has perfect information.

Observe that the pure strategies of MAX are in bijection with the subsets of S. For each subset $J \subseteq S$, if MAX plays the pure strategy corresponding to J via choosing ✓ (respectively ×) for $j \in J$ (respectively $j \notin J$), then MAX gets an expected payoff of

$$\sum_{j \in J} (\frac{1}{n} \times nj) + \sum_{j \notin J} (\frac{1}{n} \times 0) = \sum_{j \in J} j$$

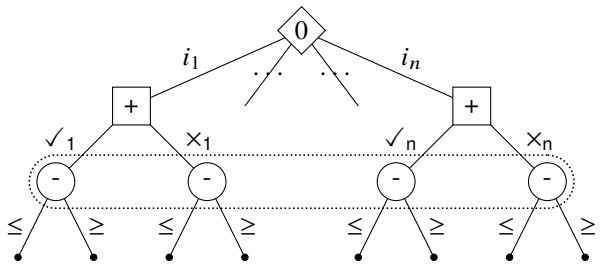


Figure 4.1: An extensive-form game that encodes an instance of Subset Sum.

against the OM $\pi_{-,\geq}$, and

$$\sum_{j\in J}(\frac{1}{n}\times(2k-nj))+\sum_{j\notin J}(\frac{1}{n}\times 2k)=2k-\sum_{j\in J}j$$

against the OM $\pi_{-,\leq}$. Hence, the OM-maxmin value is

$$\max_{J\subseteq S}\min(\sum_{j\in J}j, 2k-\sum_{j\in J}j)\leq\max_{J\subseteq S}k\leq k,$$

with equality if and only if $\sum_{j\in J} j = k$ for some $J \subseteq S$.

Therefore, the maxmin value is k if and only if there is a subset J of S that sums up to exactly k; if no such subset exists, the maxmin value is at most $k-1$. This concludes our proof of NP-hardness. Finally, NP-hardness also holds for Boolean games because one can use Lemma B.1.5 to compile the game above into a Boolean one. □

Remark. *Proposition 4.4.8 can also be proved by a reduction from the* NP*-complete problem* Knapsack*. The reduction from this problem is very similar to the one above from* Subset Sum*, but the EFG thus obtained does not depend on k, while the payoffs in the reduction above depend on k.*

4.5 Other variants of Pure Maxmin

We finally briefly discuss two natural variants of Pure Maxmin for EFGs.

Definition 4.5.1 (Pure $\leq$-Maxmin)**.** *Let $\mathcal{G}$ be a class of zero-sum EFGs. Then* Pure $\leq$-Maxmin($\mathcal{G}$) *is the following decision problem.*

Input *An EFG $G \in \mathcal{G}$ and a rational number m.*

Output *Decide whether $\underline{v}_+(\Sigma^P_+, \Sigma^P_-) \leq m$ holds in G.*

Notice that Pure $\leq$-Maxmin($\mathcal{G}$) is *not* the complement of Pure Maxmin($\mathcal{G}$), which would consist in deciding whether the maxmin value of a game is *strictly* smaller than a given threshold. Still, the complexity class of Pure $\leq$-Maxmin($\mathcal{G}$) turns out to be the complement of that of Pure Maxmin($\mathcal{G}$).

Definition 4.5.2 (Pure =-Maxmin)**.** *Let $\mathcal{G}$ be a class of zero-sum EFGs. Then* Pure =-Maxmin($\mathcal{G}$) *is the following decision problem.*

Input *An EFG $G \in \mathcal{G}$ and a rational number m.*

	No chance	Chance		
MAX \ MIN	PI/PR/MA-PR	PI	PR	MA-PR
PI	P	P	coNP-c/DP-c	Π_2^P-c/D_2^P-c
PR	P	coNP-c/DP-c	coNP-c/DP-c	Π_2^P-c/D_2^P-c
MP-PR	coNP-c/DP-c	coNP-c/DP-c	coNP-c/DP-c	Π_2^P-c/D_2^P-c

Table 4.4: Complexity of PURE ≤-MAXMIN and PURE =-MAXMIN, on the left and right of each cell, respectively.

Output *Decide whether* $\underline{v}_+(\Sigma_+^P, \Sigma_-^P) = m$ *holds in* G.

The results are summarized in Table 4.4. As it turns out, all results are parallel to those for PURE MAXMIN in Table 4.1.

Proposition 4.5.3. *The* P*-membership results in Table 4.4 hold.*

Proof. This follows from the fact that there is a polynomial-time algorithm for *computing* the maxmin value in these cases (cf. Proposition 4.2.4 and Proposition 4.2.9). □

Proposition 4.5.4. *The* coNP*-completeness and* Π_2^P*-completeness results in Table 4.4 hold. They hold even under the restrictions to* 2 *agents for MAX (in case MAX has MA-PR) and to Boolean games.*

Proof. For membership in coNP, we can check that for all pure strategies of MAX, the best response of MIN (which can be computed in linear time in these cases; see Proposition 4.2.3 and more precisely Proposition 4.4.3) yields at most m for MAX.

For membership in Π_2^P, we can check that for all pure strategies of MAX, there exists a strategy of MIN which yields at most m for MAX.

For completeness, the argument is similar for all cases. Consider for example EFGs of chance in which MAX has PI and MIN has PR. In the proof of Proposition 4.2.10, we give a reduction from 3-COLOURING to PURE MAXMIN, such that if a graph has a legal 3-colouring, the maxmin value of the EFG obtained is at least $1 - 1/|V|$, and otherwise it is at most $1 - 2/|V|$. Hence, this also gives us a reduction from the coNP-complete complement of 3-COLOURING to PURE ≤-MAXMIN with $1 - 2/|V|$ as threshold.

The other cases are similar.

- For EFGs of chance in which MAX has PR and MIN has PI (cf. Proposition 4.2.11), EFGs from "yes" instances of 3-COLOURING have a maxmin value of at least $1 - 1/|V|$ while those from "no" instances have a maxmin value of at most $1 - 2/|V|$.
- For EFGs of no chance in which MAX has MA-PR and MIN has PI (cf. Proposition 4.2.8), EFGs from "yes" instances of 3-COLOURING have a maxmin value of at least 1 while those from "no" instances have a maxmin value of at most 0.
- Finally, for EFGs of chance in which MAX has PI and MIN has MA-PR (cf. Proposition 4.2.12), EFGs from "yes" instances of TILING EXTENSION have a maxmin value of at least $1/m^4$ while those from "no" instances have a maxmin value of at most 0.

□

For $\mathcal{G}$ such that Pure Maxmin($\mathcal{G}$) is NP-complete, we have observed that Pure $\leq$-Maxmin($\mathcal{G}$) is coNP-complete, hence Pure =-Maxmin($\mathcal{G}$) is in DP[19] (since a game has a maxmin value k if and only if its maxmin value is both at least k and at most k). We show below that Pure =-Maxmin($\mathcal{G}$) is also DP-hard.

Proposition 4.5.5. *The* DP*-completeness results in Table 4.4 hold for EFGs of chance in which MAX has PI and MIN has PR, and for EFGs of chance in which MAX has PR and MIN has PI. They hold even under the restriction to Boolean games.*

Proof. The membership results are shown as discussed above. For hardness, we will give a reduction from two instances of 3-Colouring such that the EFG of chance thus obtained has a certain maxmin value if and only if the first instance of 3-Colouring is a "yes" instance and the second one is not.

Let (V_1, E_1) and (V_2, E_2) be two arbitrary graphs. Notice that adding vertices with no edge to a graph does not change its 3-colourability. Hence, we assume without loss of generality that $n \coloneqq |V_1| = |V_2|$.

Consider first the case in which MAX has PI and MIN has PR. Let G_1 (respectively G_2) be the EFG of chance built from (V_1, E_1) (respectively (V_2, E_2)) in the proof of Proposition 4.2.10. Then G_1 (respectively G_2) has a maxmin value of at least $1 - 1/n$ if (V_1, E_1) (respectively (V_2, E_2)) is 3-colourable, and at most $1 - 2/n$ otherwise.

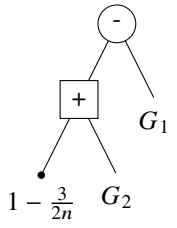


Figure 4.2: An EFG with two 3-colouring subgames G_1 and G_2.

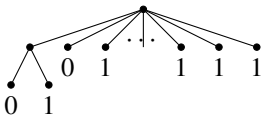


Figure 4.3: A Boolean chance tree of linear size in n representing the value $1 - 3/2n$.

Now consider the EFG G depicted in Figure 4.2: at the root, MIN chooses whether to play G_1; if they do, then the game proceeds as in G_1. Otherwise, MAX chooses whether to play the game G_2; if they do, then the game proceeds as in G_2. Otherwise, MAX receives a payoff of $1 - 3/2n$.

Since both G_1 and G_2 are EFGs of chance in which MAX has PI and MIN has PR, the game G is also an EFG of chance in which MAX has PI and MIN has PR. In addition, the construction is polynomial-time in the two graphs in input.

It remains to show that G has a maxmin value of exactly $1 - 3/2n$ if and only if (V_1, E_1) is 3-colourable while (V_2, E_2) is not. This is straightforward to verify.

- If (V_1, E_1) is not 3-colourable, then MIN can choose to play G_1 so that the maxmin value of G is at most $1 - 2/n < 1 - 3/2n$;
- If both (V_1, E_1) and (V_2, E_2) are 3-colourable, the maxmin value of G is at least $1 - 1/n > 1 - 3/2n$ since MAX can get at least $1 - 1/n$ from both G_1 and G_2.

[19] See Appendix A.2 for the definition of the class DP and its generalisation to higher levels of the polynomial hierarchy.

- If (V_1, E_1) is 3-colourable but (V_2, E_2) is not, then it is optimal for MIN to choose not to play G_1 and for MAX to choose not to play G_2. Hence, the maxmin value of G is exactly $1 - 3/2n$.

This concludes the DP-hardness of =-MAXMIN for EFGs of chance in which MAX has PI and MIN has PR. The same construction as above with Proposition 4.2.11 also shows the DP-hardness of =-MAXMIN for EFGs of chance in which MAX has PR and MIN has PI.

To show that these two problems remain DP-hard for Boolean games, we compile away the constant $1 - 3/2n$ using a technique similar to Lemma B.1.5, since G_1 and G_2 are already Boolean by construction. We give a possible compilation for this constant in Figure 4.3, which has a linear size (in n) and involves only uniform chance nodes. □

Remark. *Notice that there is nothing magic about the constant* $1 - 3/2n$ *in the proof; one can actually choose any rational number strictly between* $1 - 1/n$ *and* $1 - 2/n$ *that is expressible in a polynomial (in n) number of bits.*

Proposition 4.5.6. *The* DP*-completeness results in Table 4.4 hold for EFGs in which MAX has MA-PR. They hold even under the restriction to* 2 *agents for MAX. For EFGs of chance, they hold even under the further restriction to Boolean games.*

Proof. For EFG of no chance in which MAX has MA-PR and MIN has PI, we use the reduction in the proof of Proposition 4.2.8, which yields an EFG of no chance with 2 agents for MAX; its maxmin value is 1 if the graph is 3-colourable, and 0 otherwise.

The construction then is the same as in Figure 4.2, but the payoff of the leftmost leaf is replaced by $1/2$. By construction, the reduction is to an EFG with three values (0, $1/2$ and 1). For games of chance, the constant $1/2$ can be compiled into Boolean payoffs. □

Remark. *When chance nodes are not allowed, we cannot compile the constant* $1/2$ *into Boolean payoffs. As a result, the precise complexity of* =-MAXMIN *for Boolean EFGs of no chance in which MAX has MA-PR and MIN has PI is still an open question. We know that it is* NP*-hard since by deciding whether a Boolean game has a maxmin value of exactly* 1*, one can decide whether a graph is 3-colourable by the construction in the proof of Proposition 4.2.8; similarly, it is also* coNP*-hard.*

However, intuitively, this problem does not seem to be DP*-hard, since one* NP *oracle call is enough to decide it,*[20] *while* DP*-hard problems seem to require two such oracle calls.*

Similarly, for $\mathcal{G}$ such that PURE MAXMIN($\mathcal{G}$) is Σ_2^{P}-complete, we have observed that PURE $\leq$-MAXMIN($\mathcal{G}$) is Π_2^{P}-complete, hence PURE =-MAXMIN($\mathcal{G}$) is in $\mathsf{D}_2^{\mathsf{P}}$; we show below that it is also $\mathsf{D}_2^{\mathsf{P}}$-hard.

Proposition 4.5.7. *The* $\mathsf{D}_2^{\mathsf{P}}$*-completeness results in Table 4.4 hold. They hold even under the restriction to* 2 *agents for MIN and to Boolean games.*

Proof. The membership results are shown as discussed above. For hardness, we use a construction similar to the one in Proposition 4.5.5, but with a reduction from a pair of instances of TILING EXTENSION (cf. Proposition 4.2.12).

[20] The pure maxmin value of Boolean games of no chance is either 0 or 1, hence if the maxmin value is at least 1 (which can be known from one NP oracle call) then it must be 1; if not, then it must be 0.

Given the EFG of chance G_1 reduced from $(D_1, H_1, V_1, 1^{m_1})$ and G_2 reduced from $(D_2, H_2, V_2, 1^{m_2})$, let $m := \max(m_1, m_2)$, and let G be the game built as in Figure 4.2, but with the payoff of the leftmost leaf replaced by $1/2m^4$. Then G has a pure maxmin value of exactly $1/2m^4$ if and only if the first instance of Tiling Extension is a "yes" instance and the second one is not. To show that the problem remains hard for Boolean games, we can use Lemma B.1.5 to compile the payoffs in G into Boolean ones. □

4.6 Conclusion

We have thoroughly investigated the computational complexity of finding a lower bound on the pure maxmin value in two-player EFGs. For each degree of imperfect information (perfect information, perfect recall, multi-agent perfect recall) for MAX and MIN, and for games of no chance or of chance, we have either given a polynomial-time algorithm, or shown completeness for a certain complexity class. This allows us to have a complete landscape of this problem (Table 4.1). In addition, we have studied the complexity landscape of the same decision problem, but under two other settings: when the EFGs are defined by some compact representations (for which we have proposed a very generic definition) (Table 4.2); when MIN is known to pick strategies from a finite set known to MAX (Table 4.3). We have also studied the complexity of finding the upper bound or the exact value of pure maxmin (Table 4.4).

Some hardness results presented in this work are already known in the literature (mostly Koller and Megiddo, 1992). However, we have strengthened many of these results in different ways: by giving a simpler reduction; by restricting the degree of imperfect information or the number of strategies of a player; by restricting to Boolean payoffs; etc. We emphasise that all the hardness results in our work hold under strong restrictions: at most 2 agents for each player; only Boolean payoffs;[21] only chance nodes with uniform distribution; only one turn per agent of MAX or MIN.

We have also exhibited several previously unknown polynomial cases for EFGs of no chance: games in which MAX has perfect recall (even if MIN has only multi-agent perfect recall); games in which MAX has perfect recall and there are (an arbitrary number of) known pure opponent models; and games in which MAX has multi-agent perfect recall and there are a constant number of known pure opponent models.

Related work on the complexity of maxmin

We have focused on the complexity of pure maxmin, but not on behaviour maxmin or mixed maxmin, since the latter ones are well-studied in the literature. For reference (and comparison with pure maxmin), we give the complexity results concerning behaviour and mixed maxmin for EFGs in Table 4.5. which all come from Koller and Megiddo (1992) and Zhang et al. (2023).[22]

The complexity of Behaviour Maxmin and Mixed Maxmin for EFGs of no chance is known to coincide with the one for EFGs of chance except in one open case, which concerns EFGs of no chance and with multi-agent perfect recall. These games can also be shown to have the same complexity as EFGs of chance by the following idea: all

[21] Except for one case: the DP-hardness of =-Maxmin is not known to hold for Boolean EFGs of no chance in which MAX has MA-PR and MIN has PI; see the remark after Proposition 4.5.6.

[22] In the literature on team games (Zhang et al., 2023, for instance), behaviour maxmin and mixed maxmin are called *TME* (team maxmin equilibrium) and *TMECor*, respectively.

MAX \ MIN	PI	PR	MA-PR
PI	P	P	coNP-c
PR	P	P	coNP-c
MA-PR	NP-c	NP-c	Σ_2^P-c/Δ_2^P-c

Table 4.5: Complexity of Behaviour Maxmin and Mixed Maxmin for EFGs of chance. In the last cell, Behaviour Maxmin and Mixed Maxmin are Σ_2^P-complete and Δ_2^P-complete, respectively.

chance nodes can be replaced by decision nodes of a new agent of MAX called MAX-Nature, who is supposed to mimic Nature's behaviour; MIN can impose large penalty on MAX by challenging MAX-Nature whenever they deviate from Nature's behaviour. In the case of TMECor, MIN is also allowed to impose large penalty on MAX whenever the other agents of MAX correlate their actions to the ones of MAX-Nature.[23] We leave the rigorous and detailed construction of such gadgets to future work.

The membership results for non-polynomial cases in Table 4.5 are proved by Zhang et al. (2023, Appendix C) for the *promise problem* that consists in deciding whether the behaviour/mixed maxmin is at least m or at most $m - \varepsilon$, where m and ε are both given as input; this is to circumvent the possibility that the exact behaviour/mixed maxmin value is irrational. For the complexity of the decision problems Behaviour Maxmin and Mixed Maxmin as we have defined in this dissertation, see the work by Gimbert et al. (2020).

Notice that in some of our proofs, the same hardness also holds for behaviour maxmin rather than just for pure maxmin. For instance, in the proof of Proposition 4.2.8 (EFGs in which MAX has MA-PR and MIN has PI), behaviour strategies would not enable MAX to get a reward of 1 when the graph is not 3-colourable. The same argument is used by Koller and Megiddo (1992) to prove the NP-hardness of this case for behaviour maxmin.

Prospectives

The main perspectives for this work are to study these decision problems related to the maxmin value with finer-grained notions of complexity. In particular, we would like to investigate the complexity of these problems in the setting of parameterized complexity. Natural parameters for an EFG are, for instance, the maximal number of alternations between MAX and MIN along any branch; the branching factor; the maximal size of an information set; the number of distinct utility values. For some of these parameters, we already know that the problems will be hard under the fixed-parameter setting as well, since they are already hard for constant values of the parameters (number of alternations, number of utility values).

Finally, an important direction for future work is to investigate the link between our framework of oracle games and various classes of the game description language GDL. Drawing such connections would allow giving a complexity picture for games compactly represented in GDL (possibly with additional information, for instance a known horizon).

[23] A similar idea will be used later in the proof of Proposition 5.3.7.

Chapter 5

Combinatorial game with incomplete information

5.1 Introduction

As this dissertation is inspired by the game of Bridge, we are particularly interested in games with incomplete information, in which players do not have common knowledge about the game they play.

Organisation of the chapter

This chapter is organised as follows.

- In Section 5.2, we present the notion of games with incomplete information and show how it is related to the notion of games with imperfect information. We introduce a new subclass of games with incomplete information, which we call combinatorial games with incomplete information. CGIIs have the defining feature that their game tree has no chance node except for an initial drawing by Nature, and has only public actions.
- In Section 5.3, we thoroughly investigate the complexity of Pure Maxmin for CGIIs; these results are parallel to those for EFGs of chance, which means that CGIIs are computationally as difficult (and thus as expressive) as EFGs of chance.
- In Section 5.4, we first study the best-defence model, which is a useful assumption to simplify complicated CGIIs with a large universe. The notions of strategy fusion and non-locality are then explored to understand intuitively why such CGIIs are hard to solve. Finally, we show Ginsberg's algorithm, a depth-first search algorithm to compute the exact maxmin value of CGIIs under the best-defence model, and analyse its running time.

5.2 Games with incomplete information

We first present the notion of games with incomplete information from the literature, then introduce a new formalism for games with incomplete information that have only public actions.

5.2.1 Incomplete information in games

In this subsection, we give a quick overview of the notion of incomplete information. We then see how games with incomplete information can be modelled by EFGs with imperfect information. For a detailed and formal definition of games with incomplete information, readers can refer to Maschler et al. (2020, Chapter 9)

Incomplete information versus imperfect information

First, we emphasise that *complete* information and *perfect* information are similar but nevertheless different notions. Faliszewski et al. (2016, Section 2.4.2) summarises informally the difference between these two confusing and widely misunderstood terms; let us rephrase their words.

- Complete information describes situations in which the whole structure of a game (the number of players, the game tree, the information sets of each player, the owner of each node, the payoff for each player at each leaf node, etc.) is common knowledge among all the players of the game. Notice that all the games we have seen until now are with complete information.
- On the other hand, perfect information is a more stringent requirement than complete information. Not only the structure of the game is common knowledge, but all players have full observability and perfect recall of the history (which is essentially a record of every decision made by every player so far). In other words, players always know their exact position in the game tree when asked to make the next decision.

In summary, complete information is a weaker notion than perfect information; equivalently, *incomplete* information is a particular case of *imperfect* information.

Example. *Card games (Bridge, Poker, Hearts, etc.) are quintessential examples of games with incomplete information. In these games, the dealing of cards is usually not publicly observable; hands dealt to the players are thus not common knowledge. As a consequence, a player may be uncertain about the payoff or available actions of the other players, which can depend on their hidden hand.*

Example. *Auction is another typical example of games with incomplete information. Indeed, in many (e.g. online) auctions, players usually do not know exactly how many players participate in the same auction, nor do they know for sure how much other players value the object to be sold by auction.*

The Aumann model of incomplete information

To model incomplete information, we need to take into account not only each player's knowledge, but also their knowledge of the knowledge of the other players, and *ad infinitum*. Such an infinite hierarchy of knowledge can be captured nicely by a finite model called Aumann model of incomplete information, which we define as follows.

Definition 5.2.1 (Aumann model of incomplete information)**.** *Let U be a finite set. An* Aumann model of incomplete information *(over U) is a tuple $\langle U, P, (\mathcal{R}_i)_{i\in P}\rangle$ where P is a finite set of players, and $\mathcal{R}_i$ is an equivalence relation over U for all $i \in P$.*

The finite set U is called the *universe*, and its elements $\omega \in U$ are called *worlds*. Intuitively, a world represents a state of nature; two worlds in the same equivalence classes of a player are indistinguishable by the player. Upon such a model, the hierarchies of knowledge of the players are well-defined (Maschler et al., 2020, Section 9.1).

Example. *In Bridge or Hearts, each possible way to deal a deck of* 52 *cards to* 4 *players is considered to be a world. Two worlds are in the same equivalence class of a player if and only if they have the same hand in both worlds: since a player cannot observe the hand of the other players, they can only distinguish worlds by their own hand.*

Remark. *An Aumann model of incomplete information over U is a Kripke's model over U in which all the accessibility relations are equivalence relations. Such a model gives a well-defined semantics to the epistemic logic (Fagin et al., 1995; Huth and Ryan, 2004):*

$$\varphi ::= A \mid \bot \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid K_i\varphi,$$

where A ranges over subsets of U and i over P. The modalities K_i for $i \in P$ are naturally interpreted as knowledge operators: K_iA intuitively means player i "knows" A. This logic is widely used in epistemic game theory to give epistemic foundations to game theory (van Ditmarsch et al., 2015, Chapter 9).

If we add a commonly known (by the players) probability distribution ρ over the universe U, also called a *common prior*, we obtain an *Aumann model of incomplete information with beliefs*. Upon such a model, hierarchies of beliefs are also well-defined (Maschler et al., 2020, Section 9.2).

From now on, we simply say *Aumann model* for an arbitrary Aumann model of incomplete information with beliefs $\langle U, P, (\mathcal{R}_i)_{i\in P}, \rho\rangle$.

The Harsanyi model of games with incomplete information

Informally, a game with incomplete information is an Aumann model $\langle U, P, (\mathcal{R}_i)_{i\in P}, \rho\rangle$ together with a set of *state games* $\{G_\omega\}_{\omega\in U}$ such that for every player $i \in P$, they have the same set of strategies in two state games G_ω and $G_{\omega'}$ whenever ω and ω' are indistinguishable by them, i.e. $\omega\mathcal{R}_i\omega'$ holds.

A game with incomplete information proceeds as follows. Each player picks a strategy for every state game, with the constraint that they must pick the same strategy in two state games with worlds that are indistinguishable by them. At the beginning of the game, Nature draws a world $\omega \in U$, called the *real world*, according to the prior ρ commonly known by all players. Then, the players participate in the state game G_ω and implement their strategy in this state game.

Example. *Consider the EFG in Figure 5.1. It represents a game with incomplete information with a uniform prior over a universe of only two worlds, over which MIN has the finest equivalence class (which means MIN has complete information about the real world), while MAX has the coarsest one (which means MAX has no information about the real world). In the state game on the left, MAX and MIN play the usual Matching Pennies; in the state game on the right, the goal is reversed so that MAX aims to avoid picking the same face as MIN.*

Notice that since MAX does not know which state game they play, they have to pick either h *at all* 4 *decision nodes of them, or* t *at all* 4 *nodes. On the other hand, MIN knows the real world; hence they can choose their strategies in the two state games independently: between* H *and* T *on the left, and between* H′ *and* T′ *on the right.*

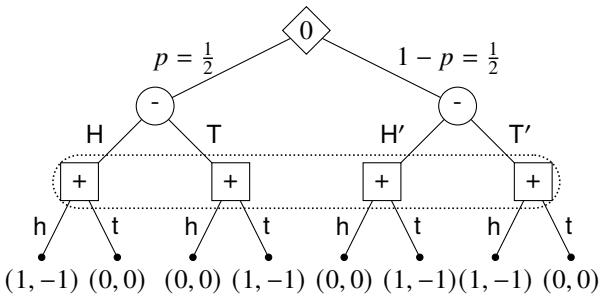


Figure 5.1: Matching Pennies with incomplete information for MAX.

The method shown in the example above, which allows modelling a game with incomplete information as an EFG with *imperfect* information, is called the *Harsanyi model of games with incomplete information*. See Maschler et al. (2020, Section 9.4) for the formal details of this model.

Therefore, throughout the remaining of this dissertation, we regard games with incomplete information as a subclass of games with imperfect information.

Ex ante vs. interim

In the literature, the Harsanyi model of a game with incomplete information is referred to as the *ex ante stage* of the game with incomplete information, while the state games are referred to as the *interim stage*. To put it in other words, the ex ante stage corresponds to the situation before the real world is drawn, or equivalently before players receive information about the state game they will play. In this dissertation, we are mostly interested in the interim stage.

5.2.2 Games with incomplete information and public actions

We propose a subclass of games with incomplete information, which contains those games with incomplete information in which state games have only public actions and no chance node. Our motivation for introducing this subclass will be explained in Subsection 5.2.3.

Definition 5.2.2 (CGII)**.** *A* combinatorial game with incomplete information *(CGII) is a tuple of the following elements:*

- *An Aumann model* $\langle U, A, (\mathcal{R}_i)_{i\in A}, \rho\rangle$ *with a finite* universe *U of* worlds*, a finite set A of* agents*, an equivalence relation* $\mathcal{R}_i$ *over U for each agent* $i \in A$*, and a* common prior $\rho \in \Delta(U)$ *over the universe;*
- *A tree T called* public tree *with nodes V and leaves* $\mathrm{L}(T)$*, the internal nodes of which are partitioned into* $(V_i)_{i\in A}$*;*
- *A reward function* $u_i : \mathrm{L}(T) \times U \to \mathbb{R}$ *for each agent* $i \in A$*.*

A CGII is said to be Boolean *if all its reward functions have values in* $\mathbb{B}$*.*

A CGII defines a game with incomplete information in which all state games have no chance node and only public actions:

- the fact that state games in a CGII have no chance node is self-evident, by the definition of the public tree of a CGII;

- the fact that they have only public actions is implicitly implied by the lack of information set description in the definition of CGII, and also explicitly implied by how we define below the notion of pure strategies in a CGII.

Remark. *In principle, we should let a CGII have a set of public trees $\{T_\omega\}_{\omega\in U}$, one for the state game in each world. However, for two-player games and maxmin values, we may assume without loss of generality that all public trees are the same one; see Subsection 5.4.2 for more details on this.*

Pure strategies in a CGII

A CGII as a game with incomplete information proceeds as follows. First, Nature randomly chooses a real world $\omega \in U$ according to ρ. Then the state game in ω proceeds from the root of the public tree. At each internal node of the public tree, the decision maker of this node chooses a successor node of the current node, depending on their equivalence class of the real world.

Concretely, the notion of strategy in a CGII is defined as follows.

Definition 5.2.3 (Pure strategy of an agent)**.** *A* pure strategy *of an agent $i \in A$ is a mapping $s_i : V_i \times U \to V$ such that*

- $\forall v \in V_i, \forall \omega \in U, s_i(v, \omega) \in \mathrm{C}(v)$*;*
- $\forall v \in V_i, \forall \omega, \omega' \in U, \omega \mathcal{R}_i \omega' \implies s_i(v, \omega) = s_i(v, \omega')$.

The set of all pure strategies of agent i is denoted by Σ_i^{P}.

Remark. *The first condition means that at each decision node of agent i, they choose a child of that node; the second condition means that agent i must choose the same child for a node in any two worlds indistinguishable by them.*

From the definition of strategy, one can see that the actions of every agent are indeed public: when making a decision at a node, an agent knows perfectly where the node is in the public tree, which in particular means they observe and remember the actions picked by every agent in the past, starting from the root of the public tree.

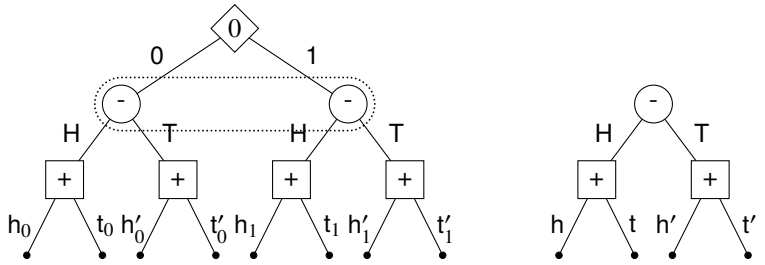


Figure 5.2: Matching Pennies with incomplete information for MIN (on the left) and its public tree (on the right), both with rewards omitted.

Example. *Consider the game with incomplete information modelled as an EFG in Figure 5.2 (on the left). This game is similar to the one in Figure 5.1, but now MAX has complete information and MIN has incomplete information. The universe can be written as* $\{\omega_0, \omega_1\}$*; MAX has the finest equivalence class and MIN has the coarsest one. The public tree of the CGII that models this game is on the right in Figure 5.2.*

On the left, since MIN does not observe the real world, they must play H *in both state games, or* T *in both. This constraint is indeed respected by the notion of strategies in a CGII: on the right, MIN only has two pure strategies* H *and* T *since* ω_0 *and* ω_1 *are indistinguishable by MIN.*

Similarly, on the left, MAX can pick between heads or tails, depending on the choices of Nature and MIN. On the right, MAX can again pick between heads or tails, depending on MIN's choice and the real world, since they can distinguish between ω_0 *and* ω_1*; note that the latter point is not reflected by the public tree, but by the Aumann model of the game.*

The notions of outcome, playout, or expected utility can be defined like for EFGs. Let $(s_1, \dots, s_n) \in \Sigma_1^{\mathrm{P}} \times \cdots \Sigma_n^{\mathrm{P}}$ be a pure strategy *profile*. We write $(s_1, \dots, s_n)(\omega)$ for the unique leaf reached under this profile when the real world is ω.

Definition 5.2.4 (Expected utility)**.** *The* expected utility *for an agent* $i \in A$ *under a pure strategy profile* $(s_1, \dots, s_n)$ *is defined to be:*

$$\mathcal{U}_i(s_1, \dots, s_n) = \sum_{\omega \in U} \rho(\omega) u_i\big((s_1, \dots, s_n)(\omega), \omega\big).$$

Teams and information in a CGII

Recall that in the literature, a *team* in an EFG is defined to be an inclusion-wise maximal set of agents with perfect recall and the same utility function. We can similarly define this notion for CGIIs.[1]

Definition 5.2.5 (Team)**.** *In a CGII, agents* i *and* j *are said to be in the same team if* $u_i = u_j$*. A* team *is an inclusion-wise maximal group of agents with the same utility function.*

We define a team's degree of incomplete information in the following way.

- *Multi-agent incomplete information* (MA-II): an arbitrary team.
- *Single-agent incomplete information* (SA-II): a team of agents with the same equivalence relation (i.e. $\mathcal{R}_i = \mathcal{R}_j$ for all agents i and j in the team).
- *Complete information* (CI): a team whose agents all have the finest equivalence relation (i.e. $\mathcal{R}_i = \{(\omega, \omega) \mid \omega \in U\}$ for all agents i in the team).

Notice that by definition, complete information implies single-agent incomplete information, which itself implies multi-agent incomplete information.

Intuitively, a team is a group of decentralised agents with shared interests working cooperatively. In a CGII, a team with single-agent incomplete information can be regarded as a team of one single agent since every agent in this team has the same information and all actions are public.

[1] The plural of "CGII" should be "CGsII", but we choose to treat CGII as a new word and write "CGIIs" instead.

Example. *In the game in Figure 5.2, team MAX (of one agent) has complete information and team MIN (of one agent) has single-agent incomplete information. If we change the owner of MIN's decision nodes to a second agent of MAX (and change their utility function accordingly so that two agents of MAX always have the same payoffs), then team MAX (now of two agents) has multi-agent incomplete information. See also Subsection 5.3.3 for more examples of CGIIs involving teams with multi-agent incomplete information.*

It should now be clear that a CGII defines a game with incomplete information (in which all state games have the same game tree with no chance node and only public actions), which itself defines an EFG of chance via the Harsanyi model (cf. Subsection 5.2.1). Due to the public actions property, there is a close link between the degree of incomplete information of a team in a CGII and the degree of imperfect information of the player corresponding to this team in the EFG defined by the CGII:

- a team with complete information in the CGII is a player with perfect information in the EFG;
- a team with single-agent incomplete information in the CGII is a player with perfect recall in the EFG;
- a team with multi-agent incomplete information in the CGII is a player with multi-agent perfect recall in the EFG.

Team maxmin in a CGII

A pure strategy of a team is uniquely defined by the pure strategy of each of its players.

Definition 5.2.6 (Pure strategy of a team)**.** *A* pure strategy of a team $\mathcal{T} \subseteq A$ *is a tuple of the form* $s_{\mathcal{T}} = (s_i)_{i \in \mathcal{T}}$ *with* $s_i \in \Sigma_i^{\mathrm{P}}$ *for all* $i \in \mathcal{T}$.

In particular, the set of pure strategies of a team $\mathcal{T}$, denoted by $\Sigma_{\mathcal{T}}^{\mathrm{P}}$, is in bijection with $\prod_{i \in A} \Sigma_i^{\mathrm{P}}$. In the following, we also write $\Sigma_{-\mathcal{T}}^{\mathrm{P}} = \prod_{i \notin \mathcal{T}} \Sigma_i^{\mathrm{P}}$, the set of all combinations of pure strategies of the players not in $\mathcal{T}$.

Definition 5.2.7 (Pure maxmin for a team)**.** *The* pure maxmin *value for a team* $\mathcal{T} \subseteq A$ *is defined to be*

$$\underline{v}_{\mathcal{T}} \coloneqq \max_{s_{\mathcal{T}} \in \Sigma_{\mathcal{T}}^{\mathrm{P}}} \min_{s_{-\mathcal{T}} \in \Sigma_{-\mathcal{T}}^{\mathrm{P}}} \mathcal{U}_{\mathcal{T}}(s_{\mathcal{T}}, s_{-\mathcal{T}}).$$

Remark. *The pure maxmin for* $\mathcal{T}$ *is well-defined since all agents in* $\mathcal{T}$ *share the same expected utility function, which we denote by* $\mathcal{U}_{\mathcal{T}}$.

Similarly, the notion of behaviour maxmin and mixed maxmin for a team[2] are also well-defined, though we do not elaborate on them here, since we mostly focus on complexity-theoretic and algorithmic aspects of pure maxmin.

In the following, we focus on zero-sum two-team CGIIs. We call the two teams *player MAX* and *player MIN*, and denote them by + and −, respectively. We emphasise again that MAX and MIN in a CGII are players with multi-agent perfect recall in the two-player EFG of chance defined by the CGII.

[2]Recall that in the literature of team games, behaviour maxmin and mixed maxmin for a team are called *TME* (team maxmin equilibrium) and *TMECor*, respectively; see Subsection 2.2.2.

5.2.3 Motivation for CGIIs

Our motivations for introducing CGII as a subclass of games with incomplete information are multiple.

Conceptual motivations

First of all, the initial idea of CGII comes from the need for a simple and minimal formalism (of subclasses of EFG games) that is capable of modelling the card play part of Bridge, the game that inspires various topics discussed in this dissertation. Recall (cf. Appendix A.4) that the card play part of Bridge is essentially a game with incomplete information in which there is one agent of MAX and two agents of MIN, and all state games have only public actions and no chance node. Hence, CGII, as it is defined in this dissertation, is a perfect fit for modelling Bridge.

Secondly, the formalism of CGIIs aims to be a minimal generalisation of that of combinatorial games to allow incomplete information.[3] Indeed, it is clear that a two-team CGII with a singleton universe is a combinatorial game. As a matter of fact, the name *combinatorial game with incomplete information* is chosen to reflect this inspiration from combinatorial games.

Lastly, but maybe most importantly, the formalism of CGII also aims to minimally capture the essence of incomplete information. Due to the public actions property, the only source of the imperfect (in particular, incomplete) information of every player comes from the initial drawing of the real world. As stated above, if the real world is common knowledge, or equivalently if the universe of a CGII is a singleton, then the game will be with perfect information and no chance (i.e. a combinatorial game).

Essence of incomplete information

Let us look more closely at this last motivation. Recall that in Chapter 4, many proofs of hardness involve constructing an EFG that has concurrent actions (among agents of the same or different teams). For instance, in the proofs of Proposition 4.2.10 (page 40) and Proposition 4.2.11 (page 40), we have essentially constructed the same game, which involves concurrent actions between MAX and MIN: MAX needs to pick a colour according to the vertex picked by Nature; MIN needs to pick a colour and an edge. By letting MAX move before MIN, we get an EFG in which MAX has PI but MIN has PR; otherwise, by letting MIN move before MAX (and Nature), we get an EFG in which MIN has PI but MAX has PR.

We believe this is in some way cheating: these two EFGs describe the same interaction between MAX and MIN, but MAX (likewise for MIN) has perfect information in one but imperfect information in the other. It is even more troubling to notice that in the EFG in which MAX has PI, MAX does not have more knowledge than they have in the other EFG, in which they only have PR. Indeed, before they make their decision in both EFGs, MAX only learns about the vertex chosen by Nature, but not about the choices of MIN.

Hence, the different degrees of imperfect information such as PI and PR do not capture the essence of incomplete information, which is a notion more about the knowledge of a player. That is why we consider games with public actions such as CGIIs. In such games, the paradox of information above, caused by different

[3] Recall that combinatorial games are two-player games of no chance and with perfect information (hence *a fortiori* public actions); see Subsection 2.2.3 for references on this subject.

chronological orders of concurrent actions, does not exist. In addition, the knowledge of the players is captured by the Aumann model and the initial drawing of the real world.

Since what interests us most is to establish a close link between the difficulty (i.e. computational complexity) of solving a game and the lack or not of knowledge (i.e. incomplete or complete information) of the players in the game, we strongly argue that CGII rather than EFG is the right model for studying sequential multi-agent interactions. If CGIIs are as hard to solve as EFGs with concurrent or hidden actions, this confirms our intuition that the difficulty of a game actually comes from the incomplete information of a player (generated by a single initial drawing over the universe), not from their inability to observe the actions chosen by the other players.

Expressiveness of CGIIs

At first sight, the requirements of no chance and public actions seem particularly restrictive: many popular tabletop games with incomplete information allow private actions (e.g. concealed Kong in Mahjong, pass in Hearts) or have randomness and chance factors beside the initial drawing (e.g. dice rolls during a game). One may worry that due to these restrictions, CGII is not expressive enough to be conceptually or algorithmically interesting. However, we argue that this impression is not correct.

First, an initial drawing over the universe is actually quite expressive. For example, for the dice rolls we evoke above, if their number and occasions are fixed in advance, then their results can be encoded into the initial drawing of worlds. Another example is given by video games, which typically use a random seed as the sole source of randomness for all procedurally generated levels and random events during a playthrough. Similar ideas have been investigated in automated planning (Palacios and Geffner, 2009, Sec. 10).

Second, even with only public actions, we will show later in Subsection 5.3.3 that we can still design a game to force a team of players to coordinate their actions. This means that we can essentially encode concurrent actions (as in standard Matching Pennies) using only public actions (and no chance except the initial drawing).

Moreover, as we shall see, CGIIs are as hard to solve as EFGs, which confirms our intuition that the difficulty of a game actually comes from the incomplete information of a player, and not from their inability to observe the moves made by the other players.

All in all, we suggest that at least as far as computation of optimal strategies is concerned, CGII, rather than EFG, be the right formalism for studying sequential multi-agent interactions depending on each player's knowledge. We will also see that many algorithmic and epistemic ideas that apply to CGII are also applicable to more general games. These reasons firmly justify our decision to focus on CGIIs in the rest of this dissertation.

5.3 Complexity of CGIIs

Parallel to EFGs (cf. Section 4.2), the decision problem Pure Maxmin for CGIIs is defined as follows.

Definition 5.3.1 (Pure Maxmin for CGIIs)**.** *Let $\mathcal{G}$ be a class of zero-sum CGIIs. Then* Pure Maxmin($\mathcal{G}$) *for CGIIs is the following decision problem.*

Input *A CGII $G \in \mathcal{G}$ and a rational number m.*

Output *Decide whether $\underline{v}_+(\Sigma_+^{\mathrm{P}}, \Sigma_-^{\mathrm{P}}) \geq m$ holds in G.*

In the following, we study the complexity of PURE MAXMIN for CGIIs in terms of different degrees of incomplete information for MAX and MIN: complete information (CI), single-agent incomplete information (SA-II), multi-agent incomplete information (MA-II). For complexity analyses, we consider the parameters $|T|$ (the number of nodes in the public tree of a CGII), $|U|$ (the number of worlds in the universe), and eventually the number of bits to encode the payoffs, the common prior, and the threshold m.

5.3.1 Summary of results

The complexity of PURE MAXMIN for CGIIs is summarised in Table 5.1. By definition, the complexity of each case is increasingly monotone both in MAX's degree of incomplete information (CI/SA-II/MA-II) and in MIN's degree of incomplete information. Hence, Table 5.1 only gives the references for the key hardness ("h") and membership ("m") results; the other results can be deduced using monotonicity. Note that results written in bold font are new from our work; the others can be directly deduced from the literature.

MAX \ MIN	CI	SA-II	MA-II
CI	P [m: 4.2.9]	**NP-c [h: 5.3.2]**	**Σ_2^{P}-c [h: 5.3.7]**
SA-II	NP-c [h: 5.3.3]	NP-c	**Σ_2^{P}-c**
MA-II	NP-c	NP-c [m: 4.2.3]	**Σ_2^{P}-c** [m: 4.2.14]

Table 5.1: Complexity of PURE MAXMIN for CGIIs.

As we have stated before, a CGII in which teams have CI (respectively SA-II or MA-II) is a game of chance in which players have perfect information (respectively perfect recall or multi-agent perfect recall). Hence, all membership results in Table 5.1 follow directly from those in Table 4.1 (page 36).

In the following, we first consider hardness results for two-agent CGIIs, i.e. CGIIs in which both MAX and MIN are single-agent (hence have either complete or single-agent incomplete information). Then we study the more complicated case in which teams are multi-agent.

5.3.2 Hardness for two-player CGIIs

Proposition 5.3.2. PURE MAXMIN *is* NP*-hard for CGIIs in which MAX has CI and MIN has SA-II. The result holds even under the restriction to Boolean games.*

Proof. We give a reduction from VERTEX COVER, which is defined as follows:

Input A non-directed graph (V, E), a natural number k.

Output Decide whether the graph has a vertex cover of size at most k.[4]

Let $\big((V, E), k\big)$ be an instance of VERTEX COVER. Without loss of generality, we assume that $k \leq |E|$, otherwise, the problem is trivial. Construct the following instance of PURE MAXMIN:

[4] A vertex cover of a graph is a subset of vertices that contains at least one endpoint of every edge of the graph.

Players MAX, who has complete information; MIN, who has single-agent incomplete information.

Aumann model The universe is $U = \{\omega_e \mid e \in E\}$. $\mathcal{R}_+$ is the finest equivalence relation, while $\mathcal{R}_-$ is the coarsest one. The common prior over the universe is the uniform one.

Public game tree The game proceeds as follows: MAX chooses a vertex $v \in V$, then MIN chooses an edge $e' \in E$.

Payoffs for MAX The payoffs are Boolean. In world ω_e, MAX gets a payoff of 1 if v is an endpoint of e and $e \neq e'$; otherwise, MAX gets 0.

Maxmin The threshold of maxmin value is $1 - \frac{k}{|E|}$.

This polynomial-time construction yields a Boolean CGII in which MAX has complete information and MIN has single-agent incomplete information. Now we show that the reduction works: the graph (V, E) has a vertex cover of size at most k if and only if the maxmin value of the constructed game is at least $1 - \frac{k}{|E|}$.

$\Longrightarrow$ Suppose first that (V, E) has a vertex cover of size at most k. Let V' be such a vertex cover. Notice that MAX can choose a vertex according to the edge e since MAX has complete information. Now consider MAX's strategies that consist in choosing v to be in the intersection of V' and the endpoints of e, which is always non-empty because V' is a vertex cover. Now we need to show that these strategies guarantee a payoff of at least $1 - \frac{k}{|E|}$.

Notice that under these strategies, only vertices in V' will ever be chosen by MAX; in other words, in no world will MIN observe that MAX picks a vertex not in V'.[5]

Since MIN does not observe e, they can only base their choice of e' on v, the vertex chosen by MAX. Intuitively, MIN needs to guess the edge chosen by the initial drawing, based on the vertex picked by MAX, to yield a reward of 0 for MAX. However, regardless of what MIN's pure strategy is, they can only guess correctly in at most one world in which MAX picks v; in other such worlds in which MIN makes a wrong guess, MAX gets 1. Since this holds for each v chosen by MAX in at least one world, which is necessarily in V', we deduce that among all $|E|$ worlds, MAX gets 0 in at most $|V'| \leq k \leq |E|$ worlds. This means the reward for such strategies is at least $1 - \frac{k}{|E|}$, therefore the maxmin value is also at least $1 - \frac{k}{|E|}$.

$\Longleftarrow$ Conversely, suppose that (V, E) has no vertex cover of size at most k, i.e. all vertex covers of the graph have size at least $k + 1$. It is clear that $|E| \geq k + 1$; otherwise, the graph must have a vertex cover of smaller size.

Notice that it is a dominating strategy for MAX to always choose a vertex v that is an endpoint of the edge e. Indeed, if MAX does not choose an endpoint of e, they get 0; if they do, they get 1 or 0, depending on MIN's choice e'. Hence, we only need to show that for all such strategies, MAX cannot guarantee a reward of at least $1 - \frac{k}{|E|}$.

Consider one such strategy, and let V' be the set of vertices that are picked by MAX in at least one world. Then by definition of this strategy, V' is a vertex cover of the graph, hence has a size of at least $k + 1$. Now, for each $v \in V'$, let MIN play an edge

[5] On the other hand, it is not necessarily the case that every vertex in V' is picked by MAX in at least one world; this is only true if V' is a minimal (by inclusion) vertex cover of the graph.

e_v which is such that MAX picks v in world ω_{e_v}. Then for each $v \in V'$, MAX gets 0 in world ω_{e_v}. Since $e_v \neq e_{v'}$ for $v \neq v'$ (by definition MAX picks v in world ω_{e_v} but v' in world $\omega_{e_{v'}}$), this means MAX gets 0 in at least $|V'| \geq k+1$ worlds; hence MAX has an expected reward of at most $1 - \frac{k+1}{|E|}$ against this strategy of MIN. This being true for all dominant strategies of MAX, the maxmin value for MAX is strictly less than $1 - \frac{k}{|E|}$. □

Remark. *Proposition 5.3.2 generalises Proposition 4.2.10 (page 40), but is more difficult to prove since we only allow games with public actions. In particular, the reduction from* 3-Colouring *in the proof of Proposition 4.2.10 no longer works, since this reduction requires that MIN be unable to observe MAX's choice of colour.*

Proposition 5.3.3. Pure Maxmin *is* NP*-hard for CGIIs in which MAX has SA-II and MIN has CI. The result holds even under the restriction to Boolean games.*

Proof. This result has been established by Frank and Basin (2001, Section 6) using a reduction from the NP-complete problem Clique, which is stated as follows:

Input A non-directed graph (V, E), a natural number k.

Output Decide whether the graph has a clique[6] of size at least k.

Here, we reproduce their proof under our formalism and fill in all the details. Let $\big((V, E), k\big)$ be an instance of Clique. Construct the following instance of Pure Maxmin:

Players MAX, who has single-agent incomplete information; MIN, who has complete information.

Aumann model The universe is $U = \{\omega_v \mid v \in V\}$. $\mathcal{R}_-$ is the finest equivalence relation, while $\mathcal{R}_+$ is the coarsest one. The common prior over the universe is the uniform one.

Public game tree The game proceeds as follows:

- At the root, MIN chooses a vertex $v' \in V$.
- MAX chooses between ✓ and ×, then the game ends.

Payoffs for MAX The payoffs are either 1 or 0. In the world ω_v:

- if MAX has chosen ✓, MAX gets a payoff of 1 if and only $(v, v') \in E$;
- otherwise, MAX gets a payoff of 1 if and only if $v \neq v'$.

Maxmin The threshold of maxmin value is $\frac{k}{n}$.

This polynomial-time construction yields a Boolean CGII in which MAX has single-agent incomplete information and MIN has complete information. Now we show that the reduction works: the graph (V, E) has a clique of size at least k if and only if the maxmin value of the constructed game is at least $\frac{k}{n}$. First, notice that MAX's pure strategies are in bijection with subsets of V: a subset $V' \subseteq V$ corresponds to MAX's pure strategy by which MAX chooses ✓ if $v' \in V'$, × otherwise; vice versa.

[6] A clique of a graph is a complete subgraph.

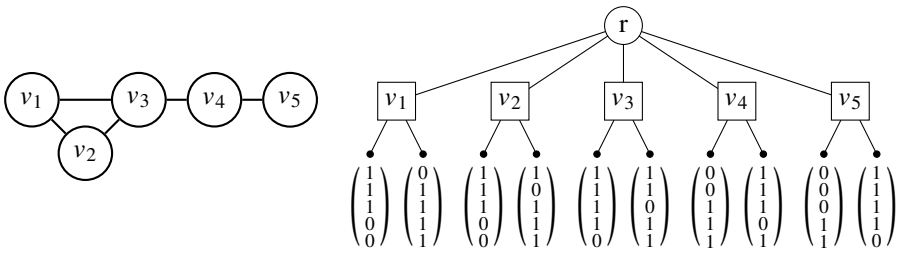


Figure 5.3: An instance of CLIQUE with k unspecified (on the left) and the Boolean CGII constructed from it (on the right), where the rewards for MAX are written in vectors in which the i-th component corresponds to MAX's reward in world ω_{v_i}. This example is also taken from the article by Frank and Basin (2001).

$\Longrightarrow$ Suppose that the graph (V, E) has a clique of size at least k, the set of vertices of which is denoted by V'. We now show that V' corresponds to a pure strategy of MAX with a payoff of at least $\frac{k}{n}$. Let $\omega_v \in U$ be the world chosen by Nature. Suppose $v \in V'$. If MIN chooses a vertex $v' \in V'$, then MAX will choose $\checkmark$, which yields a payoff of 1 since v and v are both in V', which is a clique. If MIN chooses a vertex $v' \notin V'$, then MAX will choose $\times$, which also yields a payoff 1 since $v \neq v'$. In summary, whenever the real world is ω_v with $v \in V'$ (which happens with a probability of at least $\frac{k}{n}$ since $|V'| \geq k$), regardless of what MIN chooses, MAX gets a payoff 1 under the pure strategy corresponding to V', which means this strategy has a guaranteed payoff of at least $\frac{k}{n}$.

$\Longleftarrow$ Conversely, suppose that the graph (V, E) does not have a clique of size at least k. We now show that no pure strategy of MAX can guarantee a payoff of at least $\frac{k}{n}$. Let $V' \subseteq V$ be a pure strategy of MAX. Let $V'' \subseteq V'$ be a maximal clique in the subgraph of (V, E) induced by V', then $|V''| < k$. Let $\omega_v \in U$ be the world chosen by Nature.

- Suppose $v \notin V'$. Then MIN can choose v, after which MAX will choose $\times$ under the pure strategy V' and will therefore receive a payoff of 0.
- Suppose $v \in V' \setminus V''$. Then there exists $v' \in V''$ such that $(v, v') \notin E$ (otherwise $V'' \cup \{v\}$ would be a larger clique than V'', contradicting the maximality of V''). Then MIN can choose this vertex $v' \in V'$, after which MAX will choose $\checkmark$ under the pure strategy V' and receive a payoff of 0.

Therefore, whenever $v \notin V''$ (which happens with probability $1 - \frac{|V''|}{n} > 1 - \frac{k}{n}$), MIN has a strategy such that MAX receives 0 as payoff. As a result, MAX's guaranteed payoff under the pure strategy V' is strictly smaller than $\frac{k}{n}$. □

5.3.3 Multi-agent coordination in CGIIs

Now we turn our attention to two-team CGIIs. Notice that concurrent actions, which we have used many times to show hardness results for EFGs with multi-agent perfect recall, cannot be naïvely modelled in CGIIs. Indeed, CGIIs only allow public actions, while concurrent actions require that agents in the same team be unable to observe the others' action. All hope is not lost: we show in this subsection how to construct CGIIs to impose perfect coordination between two agents in the same team, just as with concurrent actions in EFGs.

Coordination game

We first consider the following simple Boolean CGII, which we call *coordination game*. In this game, there are two agents of MAX, referred to as MAX 1 and MAX 2, and no agent of MIN; the universe has 4 worlds and reads $U = \{(b_1, b_2) \mid b_1, b_2 \in \mathbb{B}\}$; the Aumann model is such that MAX i only knows b_i for all i;[7] the common prior over U is the uniform one; the public tree is the one shown in Figure 5.4; the reward for MAX is 1 if and only if $a_1 \oplus b_1 = a_2 \oplus b_2$, where $\oplus$ is the exclusive or of two bits and a_i is the action chosen by MAX i.

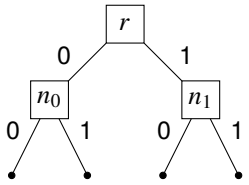


Figure 5.4: The public tree of the coordination game.

We refer to b_i as the *hidden bit* of MAX i since it is only observable by MAX i. The coordination game is designed in such a way that MAX 1 and MAX 2 must perfectly coordinate their answer in order to win. Intuitively, MAX 1 and MAX 2 need to agree on the same answer $A \in \mathbb{B}$, then stick to it during the game by playing $a_i = A \oplus b_i$. Indeed, if they employ this strategy, then they guarantee a win since

$$a_1 \oplus b_1 = (A \oplus b_1) \oplus b_1 = A = (A \oplus b_2) \oplus b_2 = a_2 \oplus b_2.$$

Remark. *Under these two winning strategies (one for each value of A), both MAX 1 and 2 pick the actions* 0 *and* 1 *with equal probability. Indeed, once the common answer A is fixed, which action to play by MAX i is dictated by their hidden bit b_i. Hence, the bits b_1 and b_2 acts as the keys of a one-time pad and perfectly*[8] encrypt/mask *the intended answer (i.e. A) of MAX 1 and 2. This is the key element to ensure that MAX 1 and 2 must cooperate without cheating.*

To show that this is the only viable winning strategy, let us consider other strategies for team MAX. First, notice that the strategies of MAX 1 can be written in the form (a_1^0, a_1^1), which means they choose a_1^0 if $b_1 = 0$ and a_1^1 if $b_1 = 1$. As for MAX 2, they have the right to pick a_2 as a function of a_1 and b_2.

- Suppose MAX 1 plays (a, a) (i.e. they always play a, regardless of what b_1 is) for $a \in \mathbb{B}$. Then MAX 2 has no winning strategy: regardless of what MAX 2 picks, they lose in at least one world: since $a \oplus 0 \neq a \oplus 1$, for every b_2, the winning condition $a_1 \oplus b_1 = a_2 \oplus b_2$ cannot be satisfied for both values of b_1.

- Suppose now MAX 1 plays (a, a') with $a \neq a'$ (hence it can also be written as $(a, a \oplus 1)$). Now, to satisfy $a_1 \oplus b_1 = a_2 \oplus b_2$, MAX 2 is forced to play $a \oplus b_2$.

Remark. *We give an intuition for these two cases:*

[7] By "only knowing b_i" we mean two worlds are in the same equivalence class of MAX i if and only if b_i has the same value in these two worlds. We will use the expression "only know", "only learn", or "only observe" with this semantics in the description of other Aumann models.

[8] In the sense of information-theoretic security (Shannon, 1949).

- *If MAX 1 cheats by using their hidden bit b_1 incorrectly (i.e. does not use b_1 to encrypt their intended answer and always picks the same action), then MAX 2 cannot cooperate perfectly since they cannot observe the value of b_1.*
- *If MAX 1 plays correctly (i.e. chooses a strategy of the form $(a, a \oplus 1)$), then MAX 2 must pick a as their intended answer and mask it with their own bit b_2 in order to win.*

Notice that in the second case, the action a_2 chosen by MAX 2 is actually probabilistically independent of the action a_1 chosen by MAX 1. This is an important feature of the coordination game, and we will return to this point shortly.

In summary, we have established the following result.

Proposition 5.3.4. *In a coordination game, team MAX guarantees a win (i.e. an expected reward of* 1*) if and only if they play a strategy of the following form: MAX i plays $A \oplus b_i$ for some $A \in \mathbb{B}$. Otherwise, their expected reward is strictly less than* 1.

Remark. *Strictly speaking, the hidden bit of MAX 2 is not necessary in a coordination game. However, this bit will be useful if we also need to mask the answer of MAX 2, e.g. in games in which MIN are not allowed to deduce information from the answer of any agent of MAX.*

Interrogation game

Consider the following situation, called *interrogation*: we have a finite set of questions Q, and MAX has a Boolean answer for each question $\{A_q \in \mathbb{B}\}_{q \in Q}$, or equivalently a mapping from Q to $\mathbb{B}$. We wish to verify whether MAX's mapping satisfies some given binary constraints $\{C_{qq'} \subseteq \mathbb{B}^2 \mid q, q' \in Q, q \neq q'\}$: MAX's mapping is said to satisfy the constraint $C_{qq'}$ if $(A_q, A'_q) \in C_{qq'}$; MAX's mapping is said to be *valid* if it satisfies all constraints.

Example. *For interrogation about 2-colourings of a given graph, the questions are the vertices of this graph; MAX's mapping is their colouring of the graph by two colours, and their answer to a vertex corresponds to the colour they assign to this vertex; the binary constraints impose the requirement that two vertices related by an edge cannot have the same colour. Concretely, $C_{vv'} = \{(0,1),(1,0)\}$ if (v, v') is an edge, otherwise $C_{vv'} = \mathbb{B}^2$. Then MAX's mapping is valid if and only if it describes a 2-colouring of the graph.*

Similarly, one can also think of interrogations about other structures over graphs, such as cliques and independent sets.

Our goal is to construct a CGII of size polynomial in $|Q|$, the number of questions, such that MAX can guarantee a win in this game if and only if there is a valid mapping from $|Q|$ to $\mathbb{B}$. Obviously, we need to verify that $(A_q, A'_q) \in C_{qq'}$ for all $q \neq q'$. However, it is not clear how we can achieve this when there is only one agent of MAX in the game[9]: we cannot question MAX about q and q' sequentially, for MAX can adapt their answer to q' based on their answer to q; neither can we ask MAX about their answers to all questions simultaneously, for it would require a game tree of size

[9] Although we will not prove it formally, such a construction of a CGII of polynomial size involving only one agent of MAX and no MIN is impossible unless P = NP, for the following reason: the pure maxmin value of such a CGII can be computed in polynomial time, while the pure maxmin value of CGIIs encoding situations of interrogation is NP-hard to compute since CLIQUE can be reduced to such an interrogation.

exponential in $|Q|$. We should exploit the presence of a second agent of MAX. Again, we cannot naïvely ask MAX 1 about q and MAX 2 about q', for the answer of MAX 1 to q is public and MAX 2 can adapt their answer to q' based on this answer. To solve this problem, we reuse the core idea of coordination games: encrypting the answer of each agent so that they cannot answer the questions adaptively.

Consider the following Boolean CGII, which we call *interrogation game*: two agents of MAX (MAX 1 and MAX 2), and no agent of MIN; the universe reads $U = \{(q_1, b_1, q_2, b_2) \mid q_1, q_2 \in Q, b_1, b_2 \in \mathbb{B}\}$; the Aumann model is such that MAX i only knows q_i and b_i for all i; the common prior over U is the uniform one; the public tree is the same one as for the coordination game (i.e. the one in Figure 5.4); the reward for MAX is 0 if either (1) $q_1 = q_2$ but $a_1 \oplus b_1 \neq a_2 \oplus b_2$[10] or (2) $q_1 \neq q_2$ but $(a_1 \oplus b_1, a_2 \oplus b_2) \notin C_{q_1 q_2}$, and 1 otherwise.

Notice that this CGII has size $O(|Q|^2)$: the universe has size $O(|Q|^2)$, while the public tree has size $O(1)$. We refer to (q_i, b_i) as the *hidden information* of MAX i. Inspired by the coordination game, we propose the following definition.

Definition 5.3.5 (Perfect coordination)**.** *In an interrogation game, a* perfect coordination *of team MAX is a pure strategy of MAX of this form: there is a set $\{A_q \in \mathbb{B}\}_{q \in Q}$ such that for all i, MAX i will play the action $a_i = A_{q_i} \oplus b_i$ in all worlds in which their hidden information is (q_i, b_i). For such a strategy, the set $\{A_q\}_{q \in Q}$ is called the* intended mapping *or* intended answer *of the perfect coordination.*

By a similar argument to the one for the coordination game, the reward condition (1) ensures that MAX 1 and 2 have an incentive to implement a perfect coordination. Otherwise, if there is a $q \in Q$ such that MAX 1 and 2 do not choose their action as in a perfect coordination, then whenever $q_1 = q_2 = q$, which happens with a non-zero probability due to the uniform prior, they will lose with a non-zero probability as shown in the argument for the coordination game. Hence, perfect coordination is a dominant strategy. In other words, (1) imposes non-adaptivity of MAX's answers.

Next, the reward condition (2) ensures that all binary constraints are satisfied by the intended mapping of a perfect coordination, since by (1) we have $a_i \oplus b_i = A_{q_i}$ for all i.

In summary, we have established the following result.

Proposition 5.3.6. *In an interrogation game, team MAX can guarantee a win (i.e. an expected reward of 1) if and only if they implement a perfect coordination and the intended mapping of this perfect coordination is valid.*

Moreover, notice that a coordination game is just an interrogation game in which there is only one question (hence no binary constraint).

Interrogation game with k-ary constraints

It is straightforward to extend the idea of interrogation games to allow constraints to be k-ary with $k \geq 2$.

Example. *For interrogation about the satisfiability of a 3-CNF, the questions are the variables appearing in the 3-CNF; MAX's mapping is their assignment to the variables; for each clause in the 3-CNF, we have a ternary constraint to impose the requirement that the assignment of MAX satisfies this clause. Then MAX's mapping is valid if and only if it is an assignment that satisfies the 3-CNF.*

[10] Recall that a_i is the action chosen by MAX i during the game.

For all fixed $k \geq 2$, the construction of an interrogation game with k-ary constraints is straightforward. Concretely, we can consider the following Boolean CGII: k agents of MAX; the universe reads $U = \{(q_1, b_1, \ldots, q_k, b_k) \mid q_1, \ldots, q_k \in Q, b_1, \ldots, b_k \in \mathbb{B}\}$; the Aumann model is such that MAX i only observes q_i and b_i for all i; the common prior over U is the uniform one; the public tree is such that each agent of MAX sequentially choose between 0 and 1; the reward for MAX is 0 if either (1) $q_i = q_j$ but $a_i \oplus b_i \neq a_j \oplus b_j$ for some i and j, or (2) $(a_1 \oplus b_2, \ldots, a_k \oplus b_k) \notin C_{q_1 \cdots q_k}$ for some $q_1, \ldots, q_k \in Q$, and 1 otherwise.

Such an interrogation game has size $O(2^k |Q|^k)$, which is still polynomial in $|Q|$ for all fixed k. This is an important feature since for an interrogation for a class of problems, $|Q|$ is usually the size of the instance (e.g. the size of a graph, of a 3-CNF formula, etc.), while k is fixed for all instances from the same class of problems (e.g. $k = 2$ for all 2-colouring problems and $k = 3$ for all 3-SAT problems).

Interrogation game for MIN

For all $k \geq 2$, one can also construct interrogation games with k-ary constraints for MIN. The construction, which now involves k agents of MIN and no agent of MAX, is very similar to the one for MAX. It suffices to modify the reward function such that the reward for MAX is 1 if either (1) $q_i = q_j$ but $a_i \oplus b_i \neq a_j \oplus b_j$ for some i and j, or (2) $(a_1 \oplus b_1, \ldots, a_k \oplus b_k) \notin C_{q_1 \cdots q_k}$ for some $q_1, \ldots, q_k \in Q$, and 0 otherwise.

In such an interrogation game, if MIN does not implement a perfect coordination, then they are caught with a probability of at least $1/(2|Q|)^2$; while if MIN implements a perfect coordination, but their intended mapping is not valid, they are caught with a probability of at least $1/|Q|^k$. Hence, if there is no valid mapping for an interrogation, MAX has a guarantee payoff of at least $\min(1/(2|Q|)^2, 1/|Q|^k) > 0$; if there is a valid mapping, then MIN can guarantee a payoff of 0 for MAX by implementing a perfect coordination for a valid mapping.

5.3.4 Hardness for two-team CGIIs

With the gadgets from the last subsection, we are ready to study the complexity of two-team CGIIs. The cases in which MAX has multi-agent incomplete information while MIN has complete or single-agent incomplete information are both NP-hard due to the NP-hardness from Proposition 5.3.3 and in NP by Proposition 4.2.3 (page 36). Therefore, they are both NP-complete. We thus focus on games in which MIN has only multi-agent incomplete information.

Our proof will be based on a reduction from the Σ_2^{P}-complete problem (Umans, 1999) Succinct Set Cover, which is defined as follows:

Input A collection[11] $S = \{\varphi_1, \ldots, \varphi_m\}$ of 3-DNF formulae on n variables, and an integer k.

Output Decide whether there exists a subcollection $S' \subseteq S$ of size at most k and the disjunction of which is a tautology: $\vee_{\varphi \in S'} \varphi \equiv 1$.

Remark. *We can duplicate the elements in the initial collection S to form a new collection $\mathcal{S}$ that contains each 3-DNF three times. It is clear that for all $1 \leq k \leq |S|$, S has a tautological disjunction of at most k of its DNFs if and only if the new collection $\mathcal{S}$ has a tautological disjunction of at most k of its DNFs. As a result, every instance*

[11] A collection, in contrast to a set, can contain two identical elements.

(S, k) of Succinct Set Cover *can be reduced to another instance $(\mathcal{S}, k)$ such that $k/|\mathcal{S}| \leq |S|/|\mathcal{S}| = 1/3$. Therefore, without loss of generality, we may suppose that all instances (S, k) of* Succinct Set Cover *satisfy $1 \leq k \leq |S|/3$.*

We first reprove Proposition 4.2.12 (page 41) with a reduction from Succinct Set Cover to illustrate the core idea of simulating finding a set cover by playing a game. Then, we will show how to augment this proof so that we obtain a CGII instead of just an EFG.

Proof redux of Proposition 4.2.12. Let (S, k) be an instance of Succinct Set Cover, where S is a collection of 3-DNF formulae on a set X of n variables and k satisfies $1 \leq k \leq |S|/3$.

Players A player MAX with perfect information, and a player MIN with multi-agent perfect recall controlling 3 agents with perfect recall, who are referred to as MIN 1, 2, and 3.

Game tree The game begins with a chance node as root and proceeds as follows:

- At the root, Nature chooses uniformly at random a 3-DNF $\varphi \in S$.
- MAX observes the choice of Nature, and chooses between $\checkmark$ and $\times$.
- Without observing the choice of Nature or MAX, MIN 1 chooses between Size, which ends the game, or Tautology.
- If MIN 1 has chosen Tautology, Nature chooses uniformly at random three variables $x_1, x_2, x_3 \in X$ and shows x_i only to MIN i.
- Each agent of MIN simultaneously chooses between 1 or 0 for the variable shown by Nature to them, and the game ends.

Payoffs for MAX MAX only receives a non-zero payoff in the following cases:

- If MIN 1 chooses Size, MAX receives 1 if they have chosen $\times$.
- If MIN 1 chooses Tautology:
  - If $x_i = x_j$ for some $i \neq j$ but MIN i and j have chosen a different assignment for this same variable, MAX receives n^2;
  - Otherwise, if MAX has chosen $\checkmark$ and a term in the 3-DNF φ is satisfied by the assignments chosen by the 3 agents of MIN, MAX receives $|S|n^3$.

Maxmin The threshold of maxmin value is $1 - k/|S|$.

The intuition behind this construction is as follows:

- MAX's pure strategies are in bijection with the subcollections of S.
- By playing Size, MIN 1 can end the game with a payoff of $1 - |S'|/|S|$ for MAX, where $S' \subseteq S$ is the pure strategy employed by MAX. This allows MIN to verify that MAX does not choose a subcollection with a size larger than k.
- If the disjunction of S' is not a tautology, then MIN 1 can play Tautology and the agents of MIN pick an assignment such that the disjunction of S' evaluates to false, which will ensure a payoff of 0 for MAX.

- The agents of MIN cannot afford to cheat by playing different assignments since they will be caught with a certain probability, which will yield a large reward for MAX.

The construction is polynomial-time in the input (S, k). Indeed, the game tree is of size $O(|S|n^3)$. In addition, MAX has perfect information, and MIN has multi-agent perfect recall. We now show that there exists a subcollection $S' \subseteq S$ of size at most k such that its disjunction is a tautology if and only if this game has a maxmin value of at least $1 - k/|S| > 0$.

$\Longrightarrow$ Suppose that there is a subcollection $S' \subseteq S$ of size at most k and the disjunction of which is a tautology. Let us consider MAX's pure strategy corresponding to S', which consists in playing $\checkmark$ if Nature chooses a 3-DNF that is in S', $\times$ otherwise.

- If MIN 1 plays Size, then MAX receives a expected payoff of $1 - |S'|/|S| \geq 1 - k/|S|$.
- Consider now the case in which MIN 1 plays Tautology. The pure strategies of each agent of MIN in the remainder of the game are in bijection with the possible assignments of X. If any two agents of MIN play different assignments, then their disagreement is caught with a probability at least $1/n^2$ (whenever a variable on which their assignment differs is chosen by Nature for both of them), which yields an expected payoff of at least 1 for MAX. Suppose now all three agents agree on an assignment. Since the disjunction of S' is a tautology, whatever the common assignment is, there is a term t from a 3-DNF φ in S' that is satisfied by this assignment. Since Nature chooses φ and the 3 variables in t with probability at least $1/(|S|n^3)$, this means MAX has an expected payoff of at least 1.

Hence, S' guarantees an expected payoff of at least $1 - k/|S|$.

$\Longleftarrow$ Conversely, suppose that there is no subcollection of S of size at most k the disjunction of which is a tautology. Let S' be an arbitrary pure strategy of MAX.

- If $|S'| > k$, then by playing Size, MIN 1 limits MAX's expected payoff to be $1 - |S'|/|S| < 1 - k/|S|$.
- Suppose now $|S'| \leq k$, then the disjunction of S' is not a tautology. Hence, there is an assignment of X such that no term in any DNF in S' is satisfied. If MIN 1 plays Tautology and then all 3 agents of MIN choose this assignment, MAX always get a payoff of $0 < 1 - k/|S|$, regardless of what Nature chooses as DNF and variables.

This being true for all $S' \subseteq S$, the maxmin value is strictly less than $1 - k/|S|$. □

Now, our goal is to prove the following, which encompasses all cases in which MIN only has multi-agent incomplete information.

Proposition 5.3.7. Pure Maxmin *is* Σ_2^{P}*-hard for CGIIs in which MAX has complete information and MIN has multi-agent incomplete information.*

We will establish this by a reduction from Succinct Set Cover. The core idea is similar to the previous reduction. However, this time we need to construct a CGII instead of an EFG, which will necessitate many complicated modifications. Hence, we

motivate every such modification in the following before presenting the lengthy proof itself.

The concurrent actions between the agents of MIN (after MIN 1 has played Tautology) can be replaced by an interrogation game gadget with 3 agents of MIN, which necessitates the introduction of three hidden bits, one for each MIN. However, this modification is not enough, for MIN can now observe MAX's choice (between ✓ and ×). This allows MIN 1 to ensure a reward of 0 for MAX by playing Size when MAX plays ✓ and Tautology when MAX plays ×.[12]

So we also need to somehow mask MAX's action, which can be achieved by granting a hidden bit b_+ to MAX. Hence, we consider a CGII with the universe

$$U = \{(\varphi, b_+, x_1, b_1, x_2, b_2, x_3, b_3) \mid \varphi \in S, x_1, x_2, x_3 \in X, b_+, b_1, b_2, b_3 \in \mathbb{B}\},$$

and an Aumann model such that MAX has complete information, and MIN i only knows x_i and b_i.

Now we have a new problem: without a second agent of MAX (unlike in an interrogation game for MAX), we cannot assure that this single agent of MAX uses b_+ correctly to encrypt their answer. In addition, the player MAX in the previous reduction from Succinct Set Cover cannot choose their action according to the variables that Nature chooses to interrogate the agents of MIN; while the player MAX in the above CGII can do this since they have complete information, and in particular they know the variables and hidden bits of the agents of MIN.

Let us write $M = (X \times \mathbb{B})^3$ and let $m = (x_1, b_1, x_2, b_2, x_3, b_3) \in M$ be an arbitrary combination of the information of the agents of MIN. We need to find a way to punish MAX so that the following two conditions are satisfied:

1. For all $\varphi \in S$ and all $b_+ \in \mathbb{B}$, the action chosen by MAX (between ✓ or ×) in a world (φ, b_+, m) depends solely on φ and b_+, not m. In other words, MAX picks the same action in (φ, b_+, m') and in (φ, b_+, m'') for all $m', m'' \in M$.

2. For all $\varphi \in S$ and $m \in M$, MAX picks a different action in $(\varphi, 0, m)$ and in $(\varphi, 1, m)$.

These conditions ensure that MAX does not (illegally) pick their answer based on their knowledge of m (condition 1), and MAX correctly encrypts their answer using their hidden bit b_+ (condition 2). Furthermore, MAX's pure strategies satisfying both conditions are in bijection with the subcollections of S. In particular, each such strategy corresponds to the subcollection

$$\{\varphi \mid \text{MAX plays } ✓ \text{ in all the worlds } (\varphi, 0, m) \text{ with } m \in M\} \subseteq S.$$

The key to ensure conditions 1 and 2 is to notice that whenever one of these conditions is violated, MIN can gain information about something (e.g. the value of φ) that they cannot observe. Hence, by adding an action to let an agent of MIN to guess which value is impossible (conditioned on MAX's action) for something that they cannot observe, MIN can punish MAX for violating these conditions. More details follow.

[12] When MAX plays ×, it suffices for the agents of MIN to pick a common assignment of the variables so that MAX always receives 0.

Ensuring condition 1

If condition 1 is violated, then there are $\varphi \in X$, $b_+ \in \mathbb{B}$, and $m', m'' \in M$, such that MAX plays $\checkmark$ in the world (φ, b_+, m') but $\times$ in the world (φ, b_+, m''). Then necessarily, $m' \neq m''$. In a world (φ, b_+, m) with an arbitrary $m \in M$, by observing MAX's action, an agent of MIN who knows φ and b_+ but not m can deduce non-trivial information about m. For example, if this agent observes $\checkmark$ from MAX, then they know for sure that $m \neq m''$, while if they observe $\times$ they know $m \neq m'$.[13]

Motivated by this observation, we introduce a new agent MIN called Controller 1, who observes φ and b_+, but not m. They have a special action called CheckM. After playing this action, they need to pick an impossible $m^* \in M$. The reward of this action will be designed such that:

- if MAX's strategy violates condition 1, then during the gameplay in at least one world, Controller 1 has an incentive to play CheckM by picking an impossible m^*, which yields a large negative expected reward for MAX;
- if MAX's strategy does not violate condition 1, then there is no impossible m^* in any world; MAX gets a large positive expected reward whenever Controller 1 plays CheckM, which means Controller 1 has no incentive to play it.

Ensuring condition 2

Now suppose that condition 1 is respected by MAX's strategy, but condition 2 is violated. Then there is $\varphi' \in S$ such that MAX plays $\checkmark$ (or $\times$) in worlds (φ', b_+, m) for all $b_+ \in \mathbb{B}$ and $m \in M$. Without loss of generality, suppose that MAX always plays $\checkmark$ in these worlds. In a world (φ, b_+, m), observing MAX playing $\times$ allows an agent of MIN who knows none of φ, b_+, and m, to conclude that $\varphi \neq \varphi'$.

Motivated by this observation, we introduce a new agent MIN called Controller 2, who observes none of φ, b_+, and m. They have a special action called CheckPhi. After playing this action, they need to pick an impossible $\varphi^* \in S$. The reward of this action will be designed such that:

- if MAX's strategy violates condition 2 by, for example, always playing $\checkmark$ for a certain φ^* regardless of what b_+ and m are, then in every world in which MAX plays $\times$, Controller 2 has an incentive to play CheckPhi by picking φ^* as an impossible value for MAX's DNF, which yields a large negative expected reward for MAX;
- if MAX's strategy does not violate condition 2, then there is no impossible φ^* in any world; MAX gets a large positive expected reward whenever Controller 2 plays CheckPhi, which means Controller 2 has no incentive to play it.

There is still a tiny loophole: what happens if MAX decides to play $\checkmark$ all the time, no matter what φ, b_+, and m are? In this case, the branch of public tree where MAX has played $\times$ is never reached, which means Controller 2 has no opportunity to play CheckPhi to punish MAX. But what does playing $\checkmark$ all the time mean for the game? MAX's action is supposed to be their answer to φ encrypted by b_+, hence when $b_+ = 0$ the action $\checkmark$ means MAX includes the DNF φ into their intended subcollection S',

[13] We recall that in the computation of maxmin values, it can be assumed that MIN has complete knowledge of the strategy of MAX.

while when $b_+ = 1$ the action ✓ means MAX does not include φ. Hence, if MIN plays Size, MAX gets an expected reward of exactly $1/2$.

However, recall that without loss of generality, we may assume that $k \leq S/3$. Hence, if S has a tautological disjunction of size at most k, then MAX can ensure an expected reward of at least $1 - k/|S| \geq 2/3$, which means MAX has no incentive to cheat by playing ✓ (and similarly for ×) in all worlds since MIN can retort by playing Size, which yields a strictly smaller reward of $1/2$. And this is the final missing piece for our proof.

Proof of Proposition 5.3.7. Let (S, k) be an instance of Succinct Set Cover, where S is a collection of 3-DNF formulae on a set X of n variables. Without loss of generality, we assume that $1 \leq k \leq |S|/3$. Construct the following instance of Pure Maxmin:

Players MAX with complete information; MIN with multi-agent incomplete information consisting of 5 agents, referred to as MIN 1, 2, 3, and Controller 1, 2.

Aumann model The universe is $U = S \times \mathbb{B} \times M$, where $M = (X \times \mathbb{B})^3$:

$$U = \{(\varphi, b_+, x_1, b_1, x_2, b_2, x_3, b_3) \mid \varphi \in S, x_1, x_2, x_3 \in X, b_+, b_1, b_2, b_3 \in \mathbb{B}\}.$$

MAX observes everything; MIN i only observes x_i and b_i for $i \in \{1, 2, 3\}$; Controller 1 only observes φ and b_+; Controller 2 observes nothing (i.e. has the coarsest equivalence relation). The common prior over the universe is the uniform one.

Public game tree Shown in Figure 5.5. MAX chooses $a_+ \in \mathbb{B}$; then Controller 1 does nothing (i.e. Pass), or plays CheckM and picks a $m^* \in M$; in the former case, Controller 2 does nothing (i.e. Pass), or plays Size, or plays CheckPhi and picks a $\varphi^* \in S$; finally, if Controller 2 does nothing, then MIN 1, 2, and 3 participate in an interrogation game with 3 agents, and they play sequentially a_1, a_2, and a_3.

Payoffs for MAX We write l_i^j, where $i \in \{1, 2, 3\}$ and $j \in \mathbb{B}$, for the leaf l_i in the subtree G_j. Let $(\varphi, b_+, m) = (\varphi, b_+, x_1, b_1, x_2, b_2, x_3, b_3)$ denote the real world, and let $N = |U|$. Then for all $j \in \mathbb{B}$:

- At l_1^j, MAX receives $+N^2$ if $m^* = m$, and $-N$ otherwise.
- At l_2^j, MAX receives $+N^2$ if $\varphi^* = \varphi$, and $-N$ otherwise.
- At l_3^j, MAX receives $1 - (a_+ \oplus b_+)$.[14]
- At each leaf of the interrogation game I^j:
  - If $x_i = x_j$ for some $i \neq j$ but $a_i \oplus b_i \neq a_j \oplus b_j$,[15] MAX receives $+N$;
  - Otherwise, if MAX has chosen $1 \oplus b_+$ and a term in the 3-DNF φ is satisfied by the assignments chosen by the 3 agents of MIN to x_1, x_2, and x_3, MAX receives $+N$.
  - Otherwise, MAX receives 0.

Maxmin The threshold of maxmin value is $1 - k/|S|$.

[14] As in the previous reduction, an intended answer of 1 means MAX includes this DNF. Furthermore, recall that $(a_+ \oplus b_+)$ represents the intended answer of MAX.

[15] This means MIN i and j intend to choose a different assignment for this same variable.

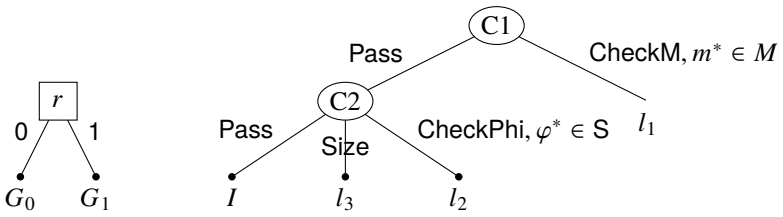


Figure 5.5: The public tree for playing Succinct Set Cover. On the right, the figure shows the subtree that repeats as G_0 and G_1 in the figure on the left. l_1, l_2, and l_3 are leaves; I is an interrogation game with three agents of MIN.

The construction above is polynomial in the size of the Succinct Set Cover instance (S, k). Indeed, the CGII has a size $O(|S|n^3)$. In addition, MAX has complete information, and MIN has multi-agent incomplete information. We now show that there exists a subcollection $S' \subseteq S$ of size at most $k \leq |S|/3$ such that its disjunction is a tautology if and only if this game has a maxmin value of at least $1 - k/|S| \geq 2/3$.

For the following arguments, it will be helpful to notice that in a world (φ, b_+, m):

- MAX picks a_+ as a function of φ, b_+, and m;
- Controller 1 picks m^* for CheckM, or plays Pass, as a function of φ, b_+, and a_+;
- Controller 2 picks φ^* for CheckPhi, or plays Pass, as a function of a_+.

$\Longrightarrow$ Suppose there is a subcollection $S' \subseteq S$ of size at most $k \leq |S|/3$ and the disjunction of which is a tautology. Let us consider MAX's pure strategy corresponding to S', which means in $(\varphi, b_+, m) \in U$, MAX plays $1 \oplus b_+$ if $\varphi \in S'$, b_+ otherwise.

We first show that MIN's dominant strategy is playing Pass by Controller 1 then Size by Controller 2 in both G_0 and G_1. This strategy of MIN yields for MAX an expected reward of $1 - |S'|/|S|$, which is at least $1 - k/|S|$ but strictly smaller than 1.

CheckM **in** G_0 **or** G_1 Let $\varphi \in S$ and $b_+ \in \mathbb{B}$, and let a_+ be MAX's action in all the worlds (φ, b_+, m) with $m \in M$, which is well-defined since a_+ depends solely on φ and b_+. If Controller 1 plays CheckM and picks $m^* \in M$ after observing φ, b_+, and a_+, then MAX receives $-N$ in all worlds (φ, b_+, m) with $m \neq m^*$, but $+N^2$ in the world (φ, b_+, m^*). Hence, the expected payoff for MAX is at least 1, which means in neither G_0 nor G_1 does Controller 1 have an incentive to play CheckM.

CheckPhi **in** G_0 **or** G_1 The argument is similar to the last one: for all $\varphi \in S$, there is at least one world in which MAX plays 0, and one in which MAX plays 1, so there is never an impossible φ^* for Controller 2 to pick. If Controller 2 plays CheckPhi, the expected payoff of MAX is at least 1. Thus, in neither G_0 nor G_1 does Controller 2 have an incentive to play CheckPhi.

I_0 **or** I_1 As shown in the previous subsection on interrogation game, in the interrogation games I_0 or I_1, the agents of MIN has an incentive to implement a perfect coordination since otherwise the discrepancy between the response of two agents of MIN will be discovered in at least one world, yielding $+N$ for MAX hence an expected reward of at least 1.[16] However, since MAX picks a subcollection S'

[16]Notice that there is no negative reward for MAX at any leaves of I.

such that the disjunction is tautological, in at least one world it will be discovered that one term from the 3-DNFs of S' is satisfied by the assignment of the agents of MIN to their variables. This again means MAX gets $+N$ in at least one world, hence an expected reward of at least 1 for the whole game. Therefore, in neither G_0 nor G_1 does Controller 2 have an incentive to choose Pass and let the game continue into an interrogation game.

In conclusion, MIN's dominant strategy is playing Pass by Controller 1 then Size by Controller 2 in both G_0 and G_1. Therefore, MAX's strategy guarantees an expected payoff of at least $1 - |S'|/|S|$, which means the maxmin value is at least $1 - k/|S|$.

$\Longleftarrow$ Suppose there is no subcollection of S of size at most $k \leq |S|/3$ such that its disjunction is a tautology. We will show that no strategy of MAX can guarantee a payoff of at least $1 - k/|S| \geq 2/3$.

Notice that by playing Pass then Size in both G_0 and G_1, MIN limits MAX's reward to be at most 1 in all worlds. We first consider the case in which MAX implements a strategy that violates one of conditions 1 and 2.

Condition 1 If MAX's strategy violates condition 1, then there are $\varphi \in S$, $b_+ \in \mathbb{B}$, and $m', m'' \in M$, such that MAX plays 1 in world (φ, b_+, m') and 0 in (φ, b_+, m''). Then in the worlds (φ, b_+, m) with $m \in M$, Controller 1 can play CheckM with $m^* = m''$ if they observe that MAX plays 1 (i.e. in G_1), and $m^* = m'$ if MAX plays 0 (i.e. in G_0). In the other worlds, Controller 1 plays Pass and Controller 2 plays Size in both G_0 and G_1. Then in no world MAX gets a reward of $+N^2$, but they get $-N$ for all worlds of the form (φ, b_+, m) and at most 1 in the other worlds. Hence, the expected world for MAX is at most $0 < 1 - k/|S|$.

Condition 2 Now suppose MAX's strategy respects condition 1 but not condition 2, then there is $\varphi' \in S$ such that MAX plays 1 (or 0) in world (φ', b_+, m) for all $b_+ \in \mathbb{B}$ and $m \in M$ (the latter being true by condition 1). Without loss of generality, suppose that MAX plays 1 in all these worlds.

- If MAX actually plays 1 in all the worlds of the universe, then Controller 1 plays Pass and Controller 2 plays Size in both G_0 and G_1 so that MAX's expected payoff is restricted to be $1/2 < 1 - k/|S|$. Indeed, with probability $1/2$, $b_+ = 0$ is drawn, yielding $a_+ \oplus b_+ = 1 \oplus b_+ = 1$, hence a payoff of 0 when Controller 2 plays Size.
- Otherwise, MAX plays 0 in at least one world in which the DNF is not φ'. Then Controller 1 plays Pass in both G_0 and G_1, Controller 2 plays Size in G_1, and CheckPhi with $\varphi^* = \varphi'$ in G_0. Then in the worlds in which MAX plays 1, they get at most 1 as reward; in all worlds (there is at least one) in which MAX plays 0, they always get $-N$ as reward since the DNF in these worlds is never φ'. Hence, the expected reward for MAX is again at most $0 < 1 - k/|S|$.

Suppose now that MAX plays a strategy that respects both conditions 1 and 2. We have already argued that such strategies are in bijection with the subcollections of S. The rest of the argument is similar to the one in the previous reduction from Succinct Set Cover:

- If MAX chooses a subcollection S' such that the disjunction is a tautology, then the size of this subcollection is at least $k + 1$. Then Controller 1 plays Pass and

Controller 2 plays Size in both G_0 and G_1 to give MAX an expected payoff of $1 - |S'|/|S| < 1 - k/|S|$.

- Otherwise, the disjunction of S' is not a tautology, then Controller 1 and 2 play Pass in both G_0 and G_1, and MIN 1, 2, and 3 implement a perfect coordination of an assignment that does not satisfy this disjunction in both of the interrogation games I_1 and I_2. As a result, MAX has an expected payoff of $0 < 1 - k/|S|$.

In conclusion, no strategy of MAX can assure a payoff of at least $1 - k/|S|$, which means the maxmin value is strictly less than $1 - k/|S|$. □

Remark. *Notice that in the CGII constructed above, MIN has* joint *complete information (i.e. if the agents of MIN could pool their information/knowledge, then they would have complete information). Indeed, Controller 1 knows φ and b_+, while MIN i knows x_i and b_i for all i. Therefore, even if we restrict our attention to CGIIs in which MIN has multi-agent incomplete information but joint complete information, Σ_2^{P}-hardness still holds. This shows the hardness comes not from MIN's inability to observe something (whatever MAX observes is observed by at least one agent of MIN), but from the fact that the agents of MIN must make decisions in a decentralised*[17] *way under different information/knowledge.*

Remark. *Note that the CGII constructed in this proof involves* 4 *different payoff values (*0*, $-N$, $+N$, and $+N^2$) and* 5 *agents of MIN. We leave to future work the search for a reduction to Boolean CGII with only* 2 *agents of MIN.*

5.3.5 Final remarks

Notice that Table 5.1 mirrors perfectly the part concerning EFGs of chance in Table 4.1 (page 36). This backs up our claim that restricting games to have only public actions and one single chance node at the beginning does not reduce the complexity of finding the pure maxmin value, which also means CGII is equally expressive as EFG, up to a polynomial factor.

The reason we did not prove results in Table 4.1 with the reductions we used in this section is a pedagogical one: due to the possibility that players' actions are private, reductions for hardness results in Table 4.1 are much more straightforward; they also give a first flavour of how games with few turns (typically one by each agent) are capable of simulating NP-hard and Σ_2^{P}-hard problems, thereby (hopefully) rendering the more sophisticated reductions in this section easier to understand.

5.4 Search algorithm for vector games

We now turn our attention to solving a CGII, or more concretely, to designing algorithms for computing the pure maxmin value of a CGII.

The case in which both teams have complete information is trivial, since such a CGII is a game of no chance and with perfect information. From Table 5.1, we know that all other cases are difficult to solve. We thus begin with the simplest non-trivial case, where MAX has single-agent incomplete information, while MIN has complete information. We also say that such a CGII is with *one-sided incomplete information*.

[17] By "decentralised", we mean that there is no central agent that collects information from each agent and informs them of what action to take.

5.4.1 The best-defence model

One-sided incomplete information is actually a case of great importance to consider due to the possibility of solving games with two-sided incomplete information by considering its approximation with one-sided incomplete information via the *best-defence model*, first introduced and formalised by Frank and Basin (1998).

The essential idea is to suppose that MIN has complete information.[18] More concretely, given a CGII G, consider a CGII G' with the same public tree, reward function, and universe, but MIN's equivalence relation over the universe is now the finest one.

By definition, MIN has no fewer equivalence classes in G' than in G, hence they have no fewer pure strategies in G' than in G. This, in return, implies that the pure maxmin value of G' is a lower bound for the one of G. In Bridge, for example, solving G' usually results in strategies that are quite close to optimal strategies in G.

What is particularly interesting about G' is that MAX's optimal strategies in it consist of optimal strategies in each equivalence class of MAX. Let us show this property by more details. By definition, the pure maxmin of G' reads

$$\underline{v}'_+ \coloneqq \max_{s_+ \in \Sigma^{\mathrm{P}}_+} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-) = \max_{s_+ \in \Sigma^{\mathrm{P}}_+} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \sum_{\omega \in U} \rho(\omega) u_+\big((s_+, s_-)(\omega), \omega\big).$$

Since MIN has complete information in G', the min over MIN's strategies can be switched with the sum over ω. Let us abuse the notation and write $\mathcal{R}_+$ the partition of the universe U induced by this equivalence relation. Let us also denote by $\Sigma^{\mathrm{P}}_{i,T}$, where $i \in \{+, -\}$ and T stands for the public tree of G', the set of all mappings from V_i to V that maps each node to one of its child[19], and denote by $s_{i,T}$ such a mapping. Notice that $\Sigma^{\mathrm{P}}_{i,T}$ is independent of the actual world. Hence, we can rewrite the pure maxmin value as

$$\underline{v}'_+ = \sum_{U' \in \mathcal{R}_+} \max_{s_{+,T} \in \Sigma^{\mathrm{P}}_{+,T}} \sum_{\omega \in U'} \rho(\omega) \min_{s_{-,T} \in \Sigma^{\mathrm{P}}_{-,T}} u_+\big((s_{+,T}, s_{-,T}), \omega\big),$$

where we denote by the tuple $(s_{+,T}, s_{-,T})$ the unique node reached under this profile.

For an equivalence class $U' \in \mathcal{R}_+$, let us consider $G'^{U'}$, the restriction of G' to U', i.e. a CGII with the same public tree and reward function as G' but with U' as its universe. In $G'^{U'}$, MAX has the coarsest equivalence relation (hence MAX has *completely* incomplete information) while MIN has the finest one (hence MIN has complete information). Then we can relate the pure maxmin value of these games by

$$\underline{v}'_+ = \sum_{U' \in \mathcal{R}_+} \rho(U') \underline{v}^{U'}_+,$$

where $\underline{v}^{U'}_+$ is the pure maxmin value of $G'^{U'}$, and $\rho(U') = \sum_{\omega \in U'} \rho(\omega)$.

As a conclusion, to compute the maxmin value of G', or equivalently to find optimal strategies in G', it is sufficient to find optimal strategies in $G'^{U'}$ for each equivalence class U' of MAX. To summarise, the best-defence model brings us from G to G', and finally to a collection of independent games $\{G'^{U'}\}_{U' \in \mathcal{R}+}$.

[18] This idea is akin to the one from Lemma 4.2.2 (page 35). Recall that for computing the maxmin value of EFGs of no chance, it can be assumed that MIN has perfect information. Of course, this result does not hold when there is at least one chance node, since the complexity of Pure Maxmin now depends on MIN's degree of imperfect information (cf. Table 5.1).

[19] Such a mapping can be considered as a pure strategy played on the public tree. A pure strategy of i in a CGII can then be regarded as mapping each equivalence class of i to such a mapping.

In typical games with incomplete information, such as card games, we are mostly interested in solving the interim stage, not the ex ante stage (cf. the end of Subsection 5.2.1). From the perspective of MAX, the interim stage under the best-defence model is just the game $G'^{U'}$ where U' is the equivalence class of MAX that contains the real world drawn by Nature.

The advantage of the best-defence model is clear: the Aumann model of $G'^{U'}$ is, in general, much simpler than that of G, and U' is also much smaller than U, which means in general solving $G'^{U'}$ for all U' is much more manageable than solving G itself. This is especially true for the reason that MIN has complete information in $G'^{U'}$ while they may only have single-agent incomplete information or multi-agent incomplete information in G ; by the complexity results in Table 5.1, we see that $G'^{U'}$ can be much easier to solve than G.

In the following, we first study how to solve a CGII under the best-defence model, i.e. solving $G'^{U'}$ for a given U' and G. Near the end of this dissertation, we will see how to go beyond the assumption of the best-defence model, while exploiting all we know about solving a CGII under this model.

5.4.2 Vector games and vectorisation

From now on, we focus on CGII with one-sided incomplete information, or equivalently CGII under the best-defence model. Since we are interested in computing the maxmin value for MAX, the utility for the other players do not matter. We may therefore, without loss of generality, assume that all CGII are zero-sum. Then the reward for MAX at each leaf can be represented by a real vector of length $|U|$ where each component signifies MAX's reward for reaching this leaf in the corresponding world. Hence, we also refer to CGII under the best-defence model as *vector games*.

Concretely, given a finite tree T, recall that we write r for its root, $\mathrm{L}(T)$ for the set of its leaves, $\mathrm{N}(T)$ for the set of its nodes, and $\mathrm{C}(n)$ for the set of children of a node $n \in \mathrm{N}(T)$. Then a vector game is formally defined as.

Definition 5.4.1 (Vector game)**.** *A* vector game*, or a* CGII under the best-defence model*, or a* CGII with one-sided incomplete information*, is a tuple* $G := \langle T, \mathrm{P}, t, \vec{u}, \vec{\rho} \rangle$*, where* T *is a finite tree,* $\mathrm{P} : \mathrm{N}(T) \setminus \mathrm{L}(T) \to \{+, -\}$ *determines whose turn it is at a node,* $t \in \mathbb{N}$ *is the number of MIN's types,* $\vec{u} : \mathrm{L}(T) \to \mathbb{R}^t$ *gives the utility for MAX depending on MIN's type, and* $\vec{\rho}$ *is a commonly known distribution over MIN's types.*

Since there is a one-to-one correspondence between MIN's types and the worlds in the universe, we use the notion "type" and "world" interchangeably. In addition, we refer to the vectors $\vec{u}(l)$ for all leaves $l \in \mathrm{L}(T)$ as *reward vectors*.

Example. *A vector game with* 5 *types of MIN is given in Figure 5.6, where a square (respectively a circle) denotes a node of MAX (respectively of MIN). We assume a uniform prior, i.e. each MIN's type is drawn with probability* $1/5$*. If MIN always plays* a *(at* r*) and MAX plays* l*, then the leftmost leaf is reached and MAX's payoff is* $(1, 1, 1, 0, 0)$*, which means MAX's gain is* 1 *if MIN is of one of the first three types, and* 0 *otherwise.*

Vectorisation

A point we have not yet addressed is that in many (not necessarily combinatorial) games with one-sided incomplete information, MIN's available actions may depend on their

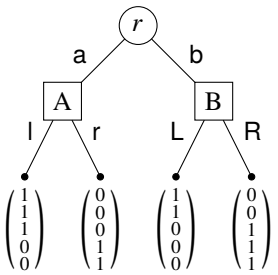


Figure 5.6: A vector game with a universe of 5 worlds.

type. For example, in a card game, MIN can only play the card in their hand. We now illustrate how to model such a game using a vector game via the idea of *vectorisation*. The core idea is to represent nodes with the same history (or, equivalently, observations) for MAX as one single node.

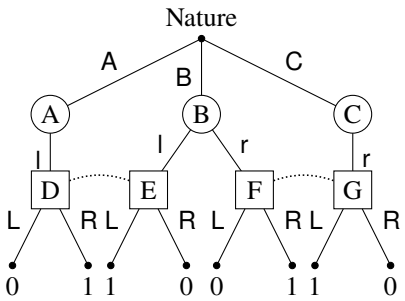


Figure 5.7: A game with one-sided incomplete information.

Take the game in Figure 5.7 as an example. Nature chooses a type among A, B and C for MIN, the available actions of whom (l, r, or both) depend on their type. On the other hand, MAX has a single type with possible actions L and R. As MAX cannot observe MIN's type, they cannot distinguish node D from node E. Therefore, these two nodes form an information set of MAX, and similarly for nodes F and G.

When we compute the maxmin value of the game, we actually view it from the perspective of MAX. In this case, any two nodes with the same history for MAX are indistinguishable. Hence, we can represent the information set $\{D, E\}$ with one single node, and similarly for the information set $\{F, G\}$. In addition, there is also no need to represent separately nodes A, B and C since they also have the same observations for MAX. Aggregating all nodes with the same history for MAX in this way yields the CGII (and vector game) in Figure 5.8.

By construction, nodes in the vector game thus obtained are in one-to-one correspondence with observations for MAX in the original game with one-sided incomplete information.

Notice that at the leftmost leaf, the payoff vector is $(0, 1, *)$. It means that the payoff of the outcome of this history is 0 if MIN is of type A, 1 if of type B. If MIN is of type C, this leaf node cannot be reached: a MIN of type C does not have the action L and therefore the history (l, L) that leads to this leaf node is impossible. This is why the third row of the payoff vector of this leaf node is denoted by an asterisk $*$.

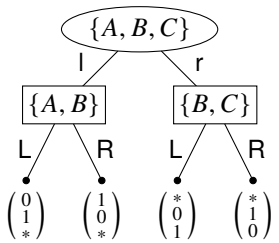


Figure 5.8: The corresponding CGII and vector game.

If we are only interested in computing the maxmin value of MAX, then all asterisks in the payoff vectors can be simply replaced by a $+\infty$, which yields a vector game with the same maxmin value of the original game, as justified by Frank and Basin (2001) and Ginsberg (2001). Indeed, MIN has perfect information, they never have an incentive to choose action that can lead to an $*$. In other words, all strategies in this vector game that contain an illegal action in the original game are weakly dominated strategies.

Following the same idea, one can also convert any two-team game with incomplete-information and public actions into a CGII with $+\infty$ and $-\infty$ featuring in some components of the reward functions with the same maxmin value.

5.4.3 Generic minimax algorithms

The maxmin value of games with perfect information is typically computed by the *minimax* algorithm, a generic version of which is shown in Algorithm 3 (the maxmin *strategies* can be computed by bookkeeping).

Algorithm 3: Generic minimax algorithm

```
def MiniMax(node n, game ⟨T, P, t, u⃗, ρ⃗⟩):
    if n is a terminal node:
        return eval(n)
    elif P(n) = +:
        return ⋁_{n′∈C(n)} MiniMax(n′)
    else:
        return ⋀_{n′∈C(n)} MiniMax(n′)
```

This algorithm has four parameters, which we use to capture different algorithms in the following.

- V is a set of objects called *situational values*;
- eval is an evaluation function which maps each leaf node $l \in \mathrm{L}(T)$ to a value $\mathsf{eval}(l) \in V$;
- $\vee, \wedge : V \times V \to V$ are two associative binary operators, referred to as MAX's and MIN's operator or *combination function*, respectively.

With eval as boundary conditions, this algorithm recursively defines a situational value $\mathsf{val}(n)$ for every node n. For an instantiation of this algorithm to compute the maxmin values, one should choose the parameters as a function of the class of games under

consideration, in such a way that there is a polynomial-time computable mapping from the situational value of the root $\mathsf{val}(r)$ to the maxmin value of the game.

We write $\texttt{MiniMax}(V, \mathsf{eval}, \vee, \wedge)$ for its instantiation with the parameters V, eval, $\vee$, $\wedge$, and denote by $\mathsf{val}(n)$ the value which it associates to each node. For example, for games with perfect information (i.e. when $t = 1$), it is well-known that $\texttt{MiniMax}(\mathbb{R}, \vec{u}, \max, \min)$ satisfies $\mathsf{val}(r) = \underline{v}_{+}$, where r is the root of the tree.

This algorithm has several advantages: returned values for internal nodes are readily interpretable; the algorithm is efficient on memory since the recursion depth is the depth of the game tree, which in general is exponentially smaller than the tree; the search can be combined with other techniques, such as heuristic functions and $\alpha\beta$ pruning (which is possible whenever $(V, \vee, \wedge)$ forms a lattice (Li et al., 2022)), move ordering, Monte Carlo techniques such as MCTS, etc.

In the following, we will show some choices of the parameters that give rise to a correct algorithm for computing the pure maxmin value of a vector game.

5.4.4 Strategy fusion and non-locality

Throughout this part, let $G = \langle T, \mathrm{P}, t, \vec{u}, \vec{\rho} \rangle$ be an arbitrary vector game. We write its universe as $U = \{\omega_1, \ldots, \omega_t\}$. Unless explicitly stated otherwise, in all our examples $\vec{\rho}$ is the uniform distribution over U.

We also assume that G is a Boolean game. This assumption is well-motivated by the following reasons.

- Boolean games are simpler; hence they simplify the discussion.
- Boolean values have a clear interpretation: 1 is a win for MAX, 0 is a loss. The situational values we will assign to nodes are easily interpretable too.
- Many tabletop games and video games, which can be modelled by games with incomplete information, have Boolean outcomes. Even when this is not the case, one can pick a threshold of score to transform a non-Boolean game into a Boolean one without losing the most important characteristics of a game.
- Last and not the least, we have seen from previous chapters on complexity results that Boolean games are as difficult and general as non-Boolean games.[20]

Under this setting, the reward vectors of the leaves (i.e. $\vec{u}(n)$ for $n \in \mathrm{L}(T)$) are binary vectors of length t, which can be regarded as subsets of the universe U. Viewing vectors as subsets allows more intuition of many subjects that will be evoked in the following. Hence, we identify elements in $\mathbb{B}^t$ with elements in $\mathcal{P}(U)$[21], and componentwise maximum (respectively, minimum) of binary vectors with union (respectively, intersection) of subsets of U. In order not to abuse the notation, we write $\|\vec{v}\| \in \mathcal{P}(U)$ for the subset of U corresponding to the binary vector $\vec{v} \in \mathbb{B}^t$.

Before giving a correct minimax algorithm for computing the pure maxmin value of a vector game G, we first show a few naïve and failed attempts in order to illustrate the particular difficulty caused by incomplete information.

[20] And one can replace $*$ by 1 in the CGII obtained by vectorisation; this allows us to get a Boolean CGII from a Boolean game.

[21] $\mathcal{P}(U)$ is the power set of U, i.e. the set of all subsets of U.

Strategy fusion

A first naïve attempt is to minimally modify the basic minimax algorithm for perfect information by assigning a binary vector of length t as situational value to each node and use component-wise maximum and minimum as combination functions. Concretely, we take $V = \mathcal{P}(U)$, $\mathsf{eval}(l) = \|\vec{u}(l)\|$ for all leaves $l \in \mathrm{L}(T)$, $\vee = \cup$, $\wedge = \cap$.

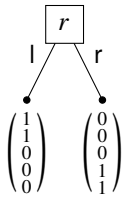


Figure 5.9: A simple vector game with a universe of 5 worlds.

Unfortunately, this approach will not work. A simple counterexample is provided by the game in Figure 5.9. In this game, MAX has two different actions l and r, while MIN does not act at all. The algorithm mentioned above will assign to the root node r a value $(1, 1, 0, 1, 1)$, which is just the component-wise maximum of the two payoff vectors $(1, 1, 0, 0, 0)$ and $(0, 0, 0, 1, 1)$. However, this result is simply wrong since it suggests that the pure maxmin value is $4/5$ while in reality it is only $2/5$ since both of MAX's strategies allow them to win in only 2 worlds.

This discrepancy is due to the fact that this naïve version of the minimax algorithm computes the value of a node owned by MAX as if they had complete information. Indeed, the component-wise maximum as the operator for MAX supposes that MAX always knows in which world they are and that they will choose the best action in this world. In our example above, the algorithm thinks that MAX can win in all worlds except world 3, which is the case if MAX can choose the correct action according to MIN's actual type. In reality, due to lack of information, MAX can only choose between l and r in a way that is independent of MIN's actual type.

This problem is called *strategy fusion* by Frank and Basin (1998). In summary, the algorithm does not take into account the incomplete information of a game; therefore, it is not what we desire in order to compute the maxmin value of a vector game. However, this algorithm, which runs only in linear time, is still useful since it computes the maxmin value of an omniscient MAX. In other words, the result yields an upper bound of the maxmin value. In addition, the vector that is the value of the root node also contains valuable information, such as whether it is possible to force a win with perfect information in a given world. For example, in the game above, it is not possible for MAX to force a win in world 3, even with complete information. Such information can provide useful heuristics and facilitate pruning in a tree search to find the real maxmin value of a game with incomplete information. We will return to this later in Subsection 6.5.2 (page 127).

Non-locality

In the last algorithm, taking the intersection as MIN's operator appears to be justifiable, since MIN has complete information and can therefore pick actions to force a loss for MAX according to the real world. As for MAX, now we have observed that taking set union as their operator does not account for MAX's single-agent incomplete

information. If we interpret the situational value of a node as the set of worlds in which MAX can force a win by picking one strategy starting from this node, then it is natural to try to change MAX's operator to pick a subset with the maximum cardinality.

However, even with this modification, the new minimax algorithm still fails. The vector game in Figure 5.6, taken from the article by Frank and Basin (1998), serves as a counterexample. Our algorithm will output $1/5$ as the pure maxmin value of this game, since it assumes that MAX greedily chooses l at node A and R at node B. Indeed, at A the action l has a support[22] of size 3 versus 2, the size of the support of the action r. Hence, the algorithm assumes that MAX chooses l at A and therefore assigns the value $(1, 1, 1, 0, 0)$ to node A. For a similar reason, it assigns the value $(0, 0, 1, 1, 1)$ to node B. Hence, at the root r, the value will be $(0, 0, 1, 0, 0)$, which is the intersection of the values of node A and node B.

Nevertheless, $1/5$ is not the correct maxmin value. In this game, MAX has in total 4 different strategies: (l, L), (l, R), (r, L), and (r, R). With a brute force computation, one can easily verify that the maxmin value of the game is, in fact, $2/5$, achieved by two maxmin strategies (l, L) (with support $\{\omega_1, \omega_2\}$) and (r, R) (with support $\{\omega_4, \omega_5\}$).

The error of this less naïve minimax algorithm is quite subtle. In a game with perfect information, the evaluation of a node n is local, in the sense that the optimal strategies in the subtree rooted at n (also called the *subgame at n*) are independent of other parts of the game tree. However, this no longer holds in the presence of incomplete information. This phenomenon is aptly named *non-locality* by Frank and Basin (1998), who also give a lengthy discussion of this phenomenon.

Let us return to the game in Figure 5.6. At node A we have action l with payoff $(1, 1, 1, 0, 0)$, which is locally better (in terms of cardinality of support) than action r with payoff $(0, 0, 0, 1, 1)$. If the real game began at node A, then l would indeed be optimal for MAX and achieve the maxmin value of $3/5$. However, it is not true that at the root of the game, it is better to prescribe action l at node A than action r since the real contribution of an action at node A to the support of a strategy at the root of the whole game tree also depends on actions chosen at other nodes than node A. For example, the fact that action l has support $\{\omega_1, \omega_2, \omega_3\}$ does not necessary mean that at the root these worlds are also in the support of an optimal strategy. Indeed, if MAX plays l at node A and R at node B, then MAX has no guarantee to win in worlds ω_1 and ω_2 since in these worlds, MIN can very well choose action b. The same is true for worlds ω_4 and ω_5, which are locally covered by action R at node B. This is the reason the strategy (l, R) only has support $\{\omega_3\}$ at the root.

In summary, in a game with incomplete information, an optimal strategy at a node x will not necessarily be part of any optimal strategy at an ancestor y of x due to non-local interactions between strategies at x and strategies at other descendants of y.

As a final remark, notice that strategy fusion is a phenomenon that occurs at MAX's nodes. On the contrary, non-locality occurs at MIN's nodes. These two phenomena illustrate different facets of MAX's incomplete information: strategy fusion is due to MAX's ignorance of the real world, while non-locality is due to their ignorance of MIN's strategies in different worlds.

5.4.5 Ginsberg's algorithm

In order to find the correct maxmin value in the example in Figure 5.6, it seems necessary to keep both l and r as options. This means that $V = \mathcal{P}(U)$ actually does not contain

[22] A *support* of a strategy is the set of worlds in which this strategy is winning.

enough information to compute the pure maxmin value at the root.

Inspired by this counterexample, one may propose $V = \mathcal{P}(\mathcal{P}(U))$. For each node n, we would like that its situational value $\mathsf{val}(n)$, a set of subsets of U (also called a *family* of U), contains information about all possible strategies of MAX in the subtree rooted at n.

Recall that each strategy of MAX in the subtree rooted at n maps each of MAX's decision nodes in this subtree to an available action at this node. In the following, we abuse the language and refer to such a strategy as a *local strategy at n*. Each strategy at n also gives rise to a vector which is its support: the set of worlds in which this strategy is winning given that the game starts at n, no matter how MIN reacts. Hence, the set of supports of all MAX's local strategies at n is a family of U. In the following, we will show how to compute this value for all nodes.

For the evaluation function, it is natural to choose $\mathsf{eval}(l) = \{\|\vec{u}(l)\|\}$ for all leaves l, since MAX has only one local strategy at l (i.e. the empty one), which has $\|\vec{u}(l)\|$ as support.

For $\vee$, we are motivated to take set union $\cup$: the set of MAX's local strategies at n is in bijection with the union of the set of MAX's local strategies at the children of n.

As for $\wedge$, we take the meet operator from the family algebra[23]:

$$\forall f, g \in \mathcal{P}(\mathcal{P}(U)), f \sqcap g = \{\alpha \cap \beta \mid \alpha \in f \wedge \beta \in g\}.$$

To justify this choice, imagine that a node z of MIN has two children x and y. Notice that the set of MAX's local strategies at z is in bijection with the product of the set of MAX's local strategies at x and y, since MAX observes MIN's action at z and can therefore choose their local strategy at x and y independently. Given a MAX's local strategy at x represented by its support f, and one at y represented by g, what will the support be for MAX's local strategy at z, denoted by (f, g), which consists in playing f at x and g at y? The answer is $f \cap g$, since MIN can choose between x and y according to the actual world so that a world is winning under MAX's local strategy (f, g) at z if and only if it is also winning under both f at x and g at y.

Hence, our new minimax algorithm reads $\texttt{MiniMax}(\mathcal{P}(\mathcal{P}(U)), \mathsf{eval}, \cup, \sqcap)$, which was first proposed by Ginsberg (2001). Consider again the game in Figure 5.6. Using this algorithm, the value for each non-leaf vertex will be:

$$\begin{aligned}
\mathsf{val}(A) &= \big\{\{\omega_1, \omega_2, \omega_3\}, \{\omega_4, \omega_5\}\big\},\\
\mathsf{val}(B) &= \big\{\{\omega_3, \omega_4, \omega_5\}, \{\omega_1, \omega_2\}\big\},\\
\mathsf{val}(r) &= \big\{\{\omega_3\}, \{\omega_1, \omega_2\}, \{\omega_4, \omega_5\}, \emptyset\big\}.
\end{aligned}$$

One can see that at the root, there are two strategies with support of size 2, hence the maxmin value is $2/5$, as we can verify by brute force.

The operators can be easily generalised to tackle non-Boolean vector games. Let us write $\mathcal{P}_{<\infty}(\mathbb{R}^t)$ for the set of all *finite* sets of vectors in $\mathbb{R}^t$. For $f, g \in \mathcal{P}_{<\infty}(\mathbb{R}^t)$, we define $f \sqcap g$ by

$$f \sqcap g \coloneqq \big\{\big(\min(v_i, v_i')\big)_{1 \le i \le t} \mid \vec{v} \in f, \vec{v}\,' \in g\big\}.$$

Proposition 5.4.2. *Let $\langle T, \mathrm{P}, t, \vec{u}, \vec{\rho}\rangle$ be a vector game with root r. For $l \in \mathrm{L}(T)$, let $\mathsf{eval}(l) \coloneqq \{\vec{u}(l)\} \in \mathcal{P}_{<\infty}(\mathbb{R}^t)$. Then the algorithm $\texttt{MiniMax}(\mathcal{P}_{<\infty}(\mathbb{R}^t), \mathsf{eval}, \cup, \sqcap)$ satisfies*

$$\underline{v}_+ \coloneqq \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \min_{s_- \in \Sigma_-^{\mathrm{P}}} u(s_+, s_-) = \max_{\vec{v} \in \mathsf{val}(r)} \vec{q} \cdot \vec{v}.$$

[23] All the operators from the family algebra are presented in Appendix A.3.1.

In other words, $\texttt{MiniMax}(\mathcal{P}_{<\infty}(\mathbb{R}^t), \mathsf{eval}, \cup, \sqcap)$ can be used to compute the pure maxmin value of a vector game. We defer the proof of this result, which is not found in the literature, to a later chapter (cf. Proposition 7.3.6, page 150) where we introduce a generalised version of this algorithm. For the moment, we content ourselves with only an intuition of how it works and why it is correct.

5.4.6 Analysis of Ginsberg's algorithm

We should emphasise that Ginsberg's algorithm $\texttt{MiniMax}(\mathcal{P}(\mathcal{P}(U)), \mathsf{eval}, \cup, \sqcap)$, where $\mathsf{eval}(l) \coloneqq \{\|\vec{u}(l)\|\}$, is *not* a polynomial-time algorithm. Recall that the size of the input is defined to be $|U| \cdot |T| = t|T|$, where $|T|$ is the number of vertices in the public tree T. It is true that this algorithm performs a depth-first search on the game tree; hence it traverses each node exactly once, which means the running time of the algorithm is linear in $|T|$ when t is fixed. On the other hand, the running time of the operators $\cup$ and $\sqcap$ are not polynomial in t.

Let us have a closer look at the computation of these operators from the family algebra in Ginsberg's algorithm. By the definition of eval, the situational value of a leaf $\mathsf{val}(l)$ is a singleton, i.e. a family containing only one subset. However, for two families f and g, $f \cup g$ is a family that has a size of $O(|f| + |g|)$, and $f \sqcap g$ is a family that has a size of $O(|f| \cdot |g|)$. This means in Ginsberg's algorithm, the cardinality of the situational values can grow exponentially when the algorithm recurses from the leaves towards the root. In the worst case, a situational value can have a cardinality of $O(2^t)$, hence computing the union $\cup$ or the meet $\sqcap$ of two such values can take a time of $O(2^t)$ (multiplied by the time needed to find the intersection of two sets of size t).

Hence, the running time of Ginsberg's algorithm is $O(2^t|T|)$, hence exponential in the input. This is hardly surprising, since we already know that finding the pure maxmin value of a vector game is NP-hard by Proposition 5.3.3. Another way to understand the exponential dependency on t is to recall that each subset in the situational value of a node represents one (or several) local strategies at this node with this subset as its support. Since MAX has exponentially more local strategies at nodes close to the root, it is natural that the situational values of these nodes have an exponential size.

What is encouraging is that this analysis also shows that Ginsberg's algorithm is fixed-parameter tractable on the size of the universe t, which means the algorithm is actually efficient when t is small (i.e. MAX does not have too much incomplete information). However, in typical card games, t is actually an astronomical number. For example, during the card play of Bridge, t is of the order of one million. In this case, it is hopeless to apply Ginsberg's algorithm directly. For this algorithm to be remotely practical, it is clearly indispensable to implement it with algorithmic optimisations, which will be the subject of the next chapter.

5.5 Conclusion

In this chapter, we have first proposed a new formalism called combinatorial game with incomplete information. This formalism models games with incomplete information with public actions, and no chance node except for the initial drawing over the universe.

We have thoroughly investigated the computational complexity of finding a lower bound on the pure maxmin in two-team CGIIs, for each degree of incomplete information (CI, SA-II, MA-II) for MAX and MIN. This allows us to have a complete landscape of this problem (cf. Table 5.1), which mirrors perfectly the complexity of EFGs of

chance (cf. Table 4.1, page 36). This shows that our new formalism CGII is as expressive as EFG of chance, but has the advantage of being conceptually simpler and more minimal, and of capturing better the essence of knowledge (or incomplete information) of players in a game.

We have then proceeded to investigate CGIIs with one-sided incomplete information (i.e. MAX has SA-II while MIN has CI). The notions of strategy fusion and non-locality have been explored to understand intuitively why such CGIIs are hard to solve. Then we have shown and analysed a correct depth-first search algorithm from the literature to compute the exact maxmin value of such CGIIs.

Prospectives

There are numerous open problems about the new formalism CGII that remain to be explored. We just list a few that are on our mind:

- Can we find a simpler proof for Proposition 5.3.7 or strengthen its result to hold even under the restrictions to 2 agents of MIN and Boolean CGIIs?
- What is the right formalism for compactly represented CGIIs? And what would be the complexity of solving them?
- What is the complexity landscape of PURE OM-MAXMIN for CGIIs? Some proofs (e.g. the one of Proposition 4.4.8, page 58) of hardness for PURE OM-MAXMIN for EFGs involve hidden or concurrent actions; hence they do not carry over to CGIIs.
- What is the complexity landscape of PURE $\leq$-MAXMIN and of PURE $=$-MAXMIN? Notice that the technique shown in Figure 4.2 (page 61) no longer works, since we cannot construct a new CGII that has two CGIIs as subtrees: chance nodes are not allowed in the public tree of a CGII.
- Can the construction of interrogation games in Subsection 5.3.3 be extended to allow questions with multi-bit answers?
- From Table 5.1, we know that CGIIs in which MAX has SA-II and MIN has CI are as hard as CGIIs in which MAX has CI and MIN has SA-II, or even as CGIIs in which both MAX and MIN have SA-II. For the first subclass, we have a depth-first exact algorithm (i.e. Ginsberg's algorithm). Can we design similar exact algorithms for the other two subclasses?

Chapter 6

Optimisations for Ginsberg's algorithm

6.1 Introduction

In this chapter, we will study various optimisations that are applicable to Ginsberg's algorithm ($\texttt{MiniMax}(\mathcal{P}(\mathcal{P}(U)), \mathsf{eval}, \cup, \sqcap)$ with $\mathsf{eval}(l) \coloneqq \{\|\vec{u}(l)\|\}$), which we roughly divide into two categories: strategy pruning and game tree pruning.

Strategy pruning Since not every local strategy (e.g. dominated strategy) can be part of a maxmin strategy at the root, it is not necessary to keep all strategies under consideration. This means we can discard certain subsets from situational values to make these families have smaller size and render the operators $\cup$ and $\sqcap$ faster to compute.

Game tree pruning Ideas such as alpha-beta pruning can be applied to Ginsberg's algorithm. In addition, there are many other possible prunings specific to games with incomplete information. These prunings allow the algorithm to avoid traversing the whole tree, and potentially reduce the size of the families generated by the algorithm due to the reduction of branching factor.

Of course, there are other optimisations. For example, by abstracting unimportant details of the opponent's hidden hands, we can reduce the number of MIN's types from millions to hundreds in a typical Bridge game. However, this particular optimisation is both *approximative*[1] and *game-specific*.

In this chapter, we are mostly interested in *exact* optimisations, i.e. the ones that guarantee that the value computed by the algorithm equipped with these optimisations is still the exact maxmin value. Although many optimisations studied in this chapter are inspired by human's gameplay techniques in Bridge, they are actually *game-agnostic* and can be applied directly to any other game modelled by vector games or CGII; some even apply to algorithms besides Ginsberg's algorithm. Hence, their presentation here will be as generic as possible to emphasise their wide applicability.

[1] Since there is no universal rule in Bridge to determine whether a small card is important or not, there is always a risk of treating two inequivalent hands of an opponent as the same.

Due to the lack of time, we defer the study of other types of optimisations (e.g. Monte Carlo tree search) to future work; also, hopefully our presentation of the formalism of CGII piques other researchers and engineers' interest in finding more optimisations.

Organisation of the chapter

This chapter is organised as follows.

- In Section 6.2, we study the notion of choice functions, which are functions that return a subset of their input. We then formalise the concept of reduction function as a particular type of choice function.
- In Section 6.3, we study the optimal generic strategy pruning for Ginsberg's algorithm, which corresponds to elimination of dominated strategies. Some other strategy prunings that are greedy but unsound, or only sound under certain circumstances, are also presented.
- In Section 6.4, we look at a very general type of tree pruning method called alpha-beta pruning. We complete the literature on this classic method by giving a characterisation of the values returned by alpha-beta search under partial order, with heuristic functions to estimate the value of a node. We also propose a caching scheme for this algorithm. Experimental results are then presented.
- In Section 6.5, we extend the idea of alpha-beta search to allow for an additional search window that we call *gamma*. After establishing the correctness of this new algorithm, we show two instances of this algorithm, which are game tree pruning methods inspired by human Bridge gameplay.

6.2 Choice functions as reduction functions

A choice function on a set S is a function that takes a non-empty subset of S and returns a non-empty subset of this subset.

Definition 6.2.1 (Choice function)**.** *Let S be an arbitrary finite set. A* choice function *on S is a function of the form $c : \mathcal{P}(S) \to \mathcal{P}(S)$ such that for all $S' \subseteq S$, $c(S') \subseteq S'$ and $c(S') \neq \emptyset$ if $S' \neq \emptyset$.*

We make a detour to study choice functions first, since many strategy prunings in Ginsberg's algorithm (e.g. elimination of dominated strategies; see Subsection 6.3.1.) are reduction functions, which are choice functions that satisfy certain properties.

In this section, we first study a few properties that a choice function can have. Then, we define reduction functions as special choice functions and provide simple sufficient conditions for a choice function to be a reduction function. Finally, we study the notion of partial reduction.

6.2.1 Properties of a choice function

The definitions from this section are inspired by the work by Moulin (1985), which readers who are willing to learn more about axiomatic approaches to choice functions should readily consult.

In the following, we focus on choice functions on $S = \mathcal{P}(O)$, where O is an arbitrary finite set. A choice function c on S is then a function defined on all families of O, the set of which we denote by $F = \mathcal{P}(\mathcal{P}(O))$.

Definition 6.2.2 (Compatibility with family algebra)**.** *Let c be a choice function on S. Then c is said to be* compatible *with family algebra (i.e. with union $\cup$, join $\sqcup$, and subset $\subseteq$) if for all $f, g \in F$:*

$$c(f \cup g) \subseteq c(f) \cup c(g), \tag{6.1}$$
$$c(f \sqcup g) \subseteq c(f) \sqcup c(g), \tag{6.2}$$
$$c(g) \subseteq c(f) \wedge f \subseteq g \implies c(f) = c(g). \tag{6.3}$$

Remark. *In the article by Moulin (1985),* (6.1) *is called* Chernoff *property;* (6.3) *is the weakened version of* Aizerman *property.*

Definition 6.2.3 (Path independence)**.** *Let c be a choice function on S. c is said to satisfy* path independence *if for all $f, g \in F$:*

$$c(f \cup g) = c(c(f) \cup c(g)), \tag{6.4}$$
$$c(f \sqcup g) = c(c(f) \sqcup c(g)). \tag{6.5}$$

Remark. *Properties* (6.4) *and* (6.5) *are referred to as "path independence" since they show that applying c to an expression with $\cup$ and $\sqcap$ is equivalent to applying it everywhere in the expression; see also Lemma 6.2.11.*

We first prove that if a choice function is compatible with family algebra, then a stronger version of (6.3) holds.

Lemma 6.2.4. *Let c be a choice function on S. If c satisfies* (6.1) *and* (6.3) *for all $f, g \in F$, then*

$$c(g) \subseteq f \subseteq g \implies c(f) = c(g). \tag{6.6}$$

Proof. Let $f, g \in F$ such that $c(g) \subseteq f \subseteq g$. Then $g = f \cup (g \setminus f)$ and by (6.1),

$$c(g) = c(f \cup (g \setminus f)) \subseteq c(f) \cup c(g \setminus f).$$

Since $c(g \setminus f) \subseteq g \setminus f$, we have

$$c(g) \cap f \subseteq \big(c(f) \cup c(g \setminus f)\big) \cap f = c(f).$$

Together with the fact that $c(g) \subseteq f$, we deduce that $c(g) = c(g) \cap f \subseteq c(f)$. Therefore, $c(g) = c(f)$ by (6.3). □

Proposition 6.2.5. *Let c be a choice function on S. Then c is compatible with family algebra if and only if c satisfies path independence.*

Proof. Suppose c is compatible with family algebra. Let $f, g \in F$. By (6.1) and (6.2),

$$c(f \cup g) \subseteq c(f) \cup c(g) \subseteq f \cup g,$$
$$c(f \sqcup g) \subseteq c(f) \sqcup c(g) \subseteq f \sqcup g.$$

Applying (6.6) yields (6.4) and (6.5). As a result, c satisfies path independence.

Suppose now c satisfies path independence. Let $f, g \in F$. Then,

$$c(f \cup g) = c(c(f) \cup c(g)) \subseteq c(f) \cup c(g),$$
$$c(f \sqcup g) = c(c(f) \sqcup c(g)) \subseteq c(f) \sqcup c(g),$$

hence (6.1) and (6.2) hold.

Suppose $c(g) \subseteq c(f)$ and $f \subseteq g$. Then by (6.4),

$$c(g) = c(f \cup g) = c(c(f) \cup c(g)) = c(c(f) \cup c(f)) = c(f),$$

which means (6.3) holds. Therefore, c is compatible with family algebra. □

Remark.

- *This result relates path independence to compatibility with family algebra. The path independence property is what we wish for a choice function, but compatibility with family algebra is usually easier to verify.*
- *In the literature (Moulin, 1985), it is known that* (6.1) *and* (6.6) *are equivalent to* (6.4)*; both are equivalent to c being pseudo-rationalisable, i.e. there is a finite set of total orders*[2] $\{\mathcal{R}_i\}_{1 \le i \le n}$ *such that for all* $f \in F$*,*

  $$c(f) = \bigcup_{1 \le i \le n} \max_f \mathcal{R}_i,$$

  which means $c(f)$ *is the set of maximum elements of* f *with respect to the total orders* $\{\mathcal{R}_i\}_{1 \le i \le n}$*.*

  Here, we imitate this result by adding the join operator $\sqcup$ *into consideration.*

6.2.2 Reduction functions

Definition 6.2.6 (Family expression)**.** *A* family expression *on* F *is an expression* v *that can be obtained by the following grammar:*

$$v ::= f \mid v_1 \cup v_2 \mid v_1 \sqcup v_2,$$

where $f \in F$*.*

In the following, we will carefully distinguish between a family expression v, which is a purely syntactic object, and its semantics, which is the value represented by the family expression v, denoted by $\|v\| \in F$.

Example. *The situational value of an internal node n in Ginsberg's algorithm is the value represented by the family expression that corresponds to the subtree rooted at n.*[3] *Concretely, for the CGII in Figure 5.6 (page 92), we may write* $v_A = f_{\mathsf{al}} \cup f_{\mathsf{ar}}$*,* $v_B = f_{\mathsf{bL}} \cup f_{\mathsf{bR}}$*,* $v_r = v_A \sqcap v_B = (f_{\mathsf{al}} \cup f_{\mathsf{ar}}) \sqcap (f_{\mathsf{bL}} \cup f_{\mathsf{bR}})$*, where* $f_{\mathsf{al}} = \{\|\vec{u}(l_{\mathsf{al}})\|\}$ *is the singleton family containing MAX's support at the leaf reached by action* a *by MIN and* l *by MAX, and similarly for the other three families. We then have* $\|v_A\| = \mathsf{val}(A) \in \mathcal{P}(\mathcal{P}(U))$*, and similarly* $\|v_B\| = \mathsf{val}(B)$*,* $\|v_r\| = \mathsf{val}(r)$*.*

[2] A *total order* is a binary relation that is transitive, antisymmetric, and complete.

[3] This is true up to a minor detail: situational values in Ginsberg's algorithm are computed with $\sqcap$ while it is $\sqcup$ that appears in family expressions; see the discussion in Subsection 6.2.4.

Definition 6.2.7 (Fully reduced family expression)**.** *Let v be a family expression and c be a choice function on S. The* fully reduced family expression $c^*(v)$ *is the expression obtained by recursively rewriting v according to the following rules:*

$$\begin{aligned} f &\to c(f), \\ v_1 \cup v_2 &\to c(c^*(v_1) \cup c^*(v_2)), \\ v_1 \sqcup v_2 &\to c(c^*(v_1) \sqcup c^*(v_2)). \end{aligned}$$

Example. *Continuing from the previous example, we have for example*

$$c^*(v_r) = c\big(c^*(v_A) \sqcap c^*(v_B)\big) = c\Big(\big(c(f_{\mathsf{al}}) \cup c(f_{\mathsf{ar}})\big) \sqcap \big(c(f_{\mathsf{bL}}) \cup c(f_{\mathsf{bR}})\big)\Big).$$

Definition 6.2.8 (Reduction function)**.** *A choice function c on S is called a* reduction function *with respect to a set Φ of weak orders*[4] *on S (represented as functions of the form $\phi : S \to \mathbb{R}$) if for all family expressions v,*

$$\forall \phi \in \Phi, \forall q \in \|v\|, \exists p \in \|c^*(v)\|, p \geq_\phi q. \tag{6.7}$$

Intuitively, each function $\phi : S \to \mathbb{R}$ expresses a preference on the elements of S. If c is a reduction function with respect to Φ, it means that to find the maximum elements in $\|v\|$ with respect to preferences in Φ, we do not have to compute $\|v\|$ itself; it is sufficient to compute $\|c^*(v)\|$, which can be obtained by recursively applying c to the family expression v in order to accelerate the computation of $\cup$ and $\sqcup$ in v.

If c is a reduction function with respect to Φ, then c must respect the preferences in Φ in the following sense.

Definition 6.2.9 (Compatibility with a set of weak-orders)**.** *A choice function c on S is said to be* compatible *with a set Φ of weak-orders on S if for all $f \in F$,*

$$\forall \phi \in \Phi, \forall q \in f, \exists p \in c(f), p \geq_\phi q.$$

Now we can show a sufficient condition for a choice function to be a reduction function.

Proposition 6.2.10. *A choice function c on S is a reduction function with respect to Φ if c is compatible with Φ and c is compatible with family algebra.*

Proof. We first show a consequence of path independence of a choice function:

Lemma 6.2.11. *A choice function c satisfies path independence if and only if $\|c^*(v)\| = c(\|v\|)$ for all family expressions v.*

Proof. The $\Longleftarrow$ direction is trivial. The $\Longrightarrow$ direction can be proved by applying properties (6.4) and (6.5) recursively. □

Suppose c is compatible with Φ and family algebra. Let v be a family expression. Since c is compatible with family algebra, by Proposition 6.2.5 c satisfies path independence, hence by Lemma 6.2.11 $\|c^*(v)\| = c(\|v\|)$. Let $\phi \in \Phi$ and $q \in \|v\|$. Since c is compatible with Φ, there exists $p \in c(\|v\|) = \|c^*(v)\|$ such that $p \geq_\phi q$. Hence, (6.7) holds and c is a reduction function with respect to Φ. □

[4] A *weak order* is a binary relation that is transitive and complete. The difference between weak order and total order is that different elements can tie under a weak order but not under a total order, since the latter requires antisymmetry.

Remark. *For c to be a reduction function with respect to Φ, the compatibilities with Φ and with family algebra are sufficient conditions as shown by this result, but also* almost *necessary in the following sense:*

- *It is clear that if c is a reduction function with respect to Φ, then c is compatible with Φ: to see this, for all families f, take v to be f in* (6.7)*.*
- *Path independence is not necessary for c to be a reduction function with respect to a fixed set Φ. However, if c does not satisfy path independence, then one can find a preference (i.e. weak order) ϕ such that c is compatible with $\{\phi\}$ but c is not a reduction function with respect to $\{\phi\}$.*[5]

6.2.3 Partial reduction

For a reduction function c, we can compute $\|c^*(v)\|$ instead of $\|v\|$. However, the computation of c can also be time-consuming. Hence, we are interested in knowing whether it is possible to not fully implement c for every occurrence of it in $\|c^*(v)\|$ but still satisfy (6.7).

Definition 6.2.12 (Approximation of a choice function)**.** *Let c be a choice function on S. An* approximation of c *is a choice function $\hat{c}$ on S such that*

$$\forall f \in F, c(f) \subseteq \hat{c}(f) \subseteq f.$$

Definition 6.2.13 (Approximation of a fully reduced family expression)**.** *Let c be a choice function on S. Let v be a family expression. An* approximation of $c^*(v)$ *is a family expression $\widehat{c^*(v)}$ obtained from $c^*(v)$ by replacing an arbitrary number of the symbol c by symbols representing arbitrary approximations of c.*

In particular, v itself can also be considered as an approximation of $c^*(v)$ in which all occurrences of c are replaced by the identity function on F. Furthermore, notice that different occurrences of the symbol c can be replaced by symbols representing different approximations of c.

Example. *Continuing the example after Definition 6.2.7. Then*

$$c_1\Big(\big(c_2(f_{\mathsf{al}}) \cup c_3(f_{\mathsf{ar}})\big) \sqcap \big(c_4(f_{\mathsf{bL}}) \cup c_5(f_{\mathsf{bR}})\big)\Big),$$

where c_i for $1 \leq i \leq 5$ are symbols for (not necessarily the same) approximations of c, is an approximation of the fully reduced family expression $c^*(v_r)$.

An approximation of c outputs something between c and the identity function. This monotonicity is preserved by an arbitrary family expression if c satisfies path independence.

Proposition 6.2.14. *Let c be a choice function on S that satisfies path independence. Then, for all family expressions v and all approximations $\widehat{c^*(v)}$ of $c^*(v)$,*

$$\|c^*(v)\| \subseteq \|\widehat{c^*(v)}\| \subseteq \|v\|. \tag{6.8}$$

[5]For example, if for some $f, g \in F$, $c(f \cup g) \not\subseteq c(f) \cup c(g)$, then consider a ϕ such that all elements that are maximum in $c(f \cup g)$ with respect to ϕ are in $c(f \cup g) \setminus (c(f) \cup c(g))$.

Proof. We will prove this statement by strong induction.

Let v be a family expression of depth[6] 0 and $\widehat{c^*(v)}$ be an approximation of $c^*(v)$. Then $v = f$ with $f \in F$ and $\widehat{c^*(v)} = \hat{c}(f)$ where $\hat{c}$ is an approximation of c. Hence, (6.8) holds by the definition of an approximation of a choice function.

Now suppose the statement holds for all family expressions of depth at most $n \in \mathbb{N}$ and all their approximations. Let v be a family expression of depth $n+1$. Then $v = v_1 \cup v_2$ or $v = v_1 \sqcup v_2$, where v_1 and v_2 are two family expressions of depth at most n.

We only consider the first case, since the argument for the second case is identical. In this case, $c^*(v) = c(c^*(v_1) \cup c^*(v_2))$. Let $\widehat{c^*(v)}$ be an approximation of $c^*(v)$. Then $\widehat{c^*(v)}$ can be written as $\hat{c}(\widehat{c^*(v_1)} \cup \widehat{c^*(v_2)})$, where $\hat{c}$ is an approximation of c, $\widehat{c^*(v_1)}$ and $\widehat{c^*(v_2)}$ are respectively an approximation of $c^*(v_1)$ and $c^*(v_2)$.

Our goal is to establish (6.8). By the induction hypothesis, we have $\|c^*(v_i)\| \subseteq \|\widehat{c^*(v_i)}\| \subseteq \|v_i\|$ for $i \in \{1,2\}$. Hence,

$$\|\widehat{c^*(v)}\| = \hat{c}(\|\widehat{c^*(v_1)}\| \cup \|\widehat{c^*(v_2)}\|) \subseteq \|\widehat{c^*(v_1)}\| \cup \|\widehat{c^*(v_2)}\| \subseteq \|v_1\| \cup \|v_2\| = \|v\|,$$

which is the second part of (6.8).

To prove the first part of (6.8), first notice that since c satisfies path independence, $c(\|v_i\|) = \|c^*(v_i)\|$ for $i \in \{1,2\}$ by Lemma 6.2.11. By the induction hypothesis,

$$c(\|v_1\|) \cup c(\|v_2\|) = \|c^*(v_1)\| \cup \|c^*(v_2)\| \subseteq \|\widehat{c^*(v_1)}\| \cup \|\widehat{c^*(v_2)}\| \subseteq \|v_1\| \cup \|v_2\|.$$

Also, by path independence, c satisfies (6.1), which means

$$c(\|v_1\| \cup \|v_2\|) \subseteq c(\|v_1\|) \cup c(\|v_2\|).$$

Applying (6.6) to $f = \|\widehat{c^*(v_1)}\| \cup \|\widehat{c^*(v_2)}\|$ and $g = \|v_1\| \cup \|v_2\|$ yields

$$c(\|v_1\| \cup \|v_2\|) = c(\|\widehat{c^*(v_1)}\| \cup \|\widehat{c^*(v_2)}\|).$$

Therefore,

$$\|c^*(v)\| = c(\|v\|) = c(\|\widehat{c^*(v_1)}\| \cup \|\widehat{c^*(v_2)}\|) \subseteq \hat{c}(\|\widehat{c^*(v_1)}\| \cup \|\widehat{c^*(v_2)}\|) = \|\widehat{c^*(v)}\|,$$

which is the first part of (6.8), thus concluding the induction. □

As a consequence, any approximation of a reduction function can be used to partially reduce family expressions without sacrificing optimality. More formally.

Corollary 6.2.15. *Let c be a choice function on S that is compatible with Φ and family algebra. Then for all family expressions v and all approximations $\widehat{c^*(v)}$ of $c^*(v)$,*

$$\forall \phi \in \Phi, \forall q \in \|v\|, \exists p \in \|\widehat{c^*(v)}\|, p \geq_\phi q.$$

Proof. This is a direct result from Proposition 6.2.10 and Proposition 6.2.14. □

[6]The depth of a family expression is defined similarly to the depth of a Boolean expression

6.2.4 Conclusion

We have shown that a choice function that is compatible with family algebra and a set of weak-orders Φ can be used as a reduction function so that a fully reduced family expression $c^*(v)$ preserves the maximum values in v with respect to the preferences in Φ. Moreover, every partially reduced family, i.e. an approximation $\widehat{c^*(v)}$, also preserves this optimality.

These results give us more freedom in the implementation of c. For example, we can have an online algorithm $\hat{c}$ that computes an approximation of c such that $\hat{c}$ prioritises discarding elements from $f \in F$ that can be easily proven non-optimal, and $\hat{c}$ computes exactly $c(f)$ only when there is enough time. Likewise, we can also have a lazy version, which only applies c to a sub-expression when it is explicitly told to do so by the tree search algorithm.

We will apply the notion of reduction function to Ginsberg's algorithm. As a caveat, throughout this section we use union $\cup$ and join $\sqcup$ from family algebra, not union and meet $\sqcap$ as in Ginsberg's algorithm. However, this does not pose any technical difficulty, since in Ginsberg's algorithm, instead of computing supports for local strategies, we may compute losing supports for them, i.e. the set of worlds in which a local strategy loses. In that case, Ginsberg's algorithm becomes $\texttt{MiniMax}(\mathcal{P}(\mathcal{P}(U)), \mathsf{eval}, \cup, \sqcup)$ with $\mathsf{eval}(l) \coloneqq \{\overline{\|\vec{u}(l)\|}\}$, where $\overline{\|\vec{u}(l)\|}$ is the complement of the set $\|\vec{u}(l)\|$ in U. In the following, we ignore this detail and consider that reduction functions are defined with respect to $\sqcap$ in the case of Ginsberg's algorithm.

The applicability of reduction functions (and theirs approximations), however, is beyond this particular algorithm. For instance, Dasgupta et al. (1996) make use of reduction functions on the set of all finite subsets of real vectors of the same fixed length with respect to one weak order, in the setting of game tree search for multi-objective games.

As a matter of fact, both Ginsberg's algorithm and multi-objective game tree search by Dasgupta et al. (1996) can be unified in the same framework, akin to nondeterministic planning (Rintanen, 2004; Brafman et al., 2013). We leave the complete formalism of this framework that we call *nondeterministic game* to future work.

6.3 Strategy prunings

Equipped with the notion of reduction function, we are ready to talk about strategy prunings in Ginsberg's algorithm, which aim to discard vectors from situational values computed by Ginsberg's algorithm without modifying the maxmin value computed. Generally speaking, we are interested in choice functions on $S = \mathcal{P}(U)$ such that we can compute the pure maxmin value from $c(\mathsf{val}(r))$. If this is the case, then applying c to Ginsberg's algorithm is a *sound* strategy pruning method.

6.3.1 Elimination of dominated strategies in Ginsberg's algorithm

Recall that local strategies at a node n are implicitly represented by their support in the situational value of n. If the support of a local strategy s_+ at n is included in the support of another different local strategy s'_+ at n, then we can say that s_+ is *dominated* by s'_+, in the usual game-theoretic sense of dominance. Indeed, it will do MAX no harm to switch their local strategy at n from s_+ to s'_+, since if n is ever reached during the gameplay, then a win under s_+ is also a win under s'_+.

In particular, ignoring dominated local strategies will not change the maxmin value. This motivates us to consider the following choice function on $\mathcal{P}(U)$: discarding all dominated local strategies from all situational values. Concretely, we use the *maximal elements* operator from the family algebra, which is defined by

$$\forall f \in \mathcal{P}(\mathcal{P}(U)), f^{\uparrow} = \{p \in f \mid \forall q \in f, p \subseteq q \implies p = q\} \subseteq f.$$

Notice that this operator prunes all dominated subsets from a family so that after applying the choice function, the result will be an antichain, i.e. a set of elements that are pairwise incomparable in terms of set inclusion.

It is straightforward to verify that the operator $\uparrow$ is a reduction function in the sense of Definition 6.2.8 with respect to the set of preferences $\Phi = \{\phi_{U'}\}_{U' \subseteq U}$ where $\phi_{U'}(U'') = 1$ if $U' \subseteq U''$ and 0 otherwise.[7] Hence, we can use the operator $\uparrow$ or its approximations (which correspond to a partial elimination of dominated strategies) as a sound strategy pruning method in Ginsberg's algorithm.

Consider union and meet augmented by this reduction function:

$$f \cup^* g \coloneqq (f \cup g)^{\uparrow},$$
$$f \sqcap^* g \coloneqq (f \sqcap g)^{\uparrow},$$

where $f, g \in \mathcal{P}(\mathcal{P}(U))$ are two arbitrary families. In the following, we refer to Ginsberg's algorithm with $\uparrow$ as reduction function, i.e. $\texttt{MiniMax}(\mathcal{P}(\mathcal{P}(U)), \mathsf{eval}, \cup^*, \sqcap^*)$, as *Ginsberg's algorithm with reduction*.

As a parenthetical note, Ginsberg (2001) shows that the set of reduced families

$$V = \mathcal{P}(\mathcal{P}(U))^{\uparrow} = \{f \in \mathcal{P}(\mathcal{P}(U)) \mid f^{\uparrow} = f\}$$

together with these two binary operators $\cup^*$ and $\sqcap^*$ form a distributive lattice $(V, \cup^*, \sqcap^*)$ with the partial order $\preceq$ on V defined as

$$\forall f, g \in V, f \preceq g \iff \forall p \in f, \exists q \in g, p \subseteq q.$$

This partial order is quite natural to understand: a family of strategies g is at least as advantageous for MAX as another family of strategies f if and only if any strategy in f is dominated by at least one strategy in g. This lattice structure turns out to be critical for Ginsberg's algorithm, since it makes alpha-beta pruning sound for the algorithm. We will explore this subject thoroughly in Section 6.4.

6.3.2 Non-locality and its implications on sound strategy prunings

Optimality of $\uparrow$

We have observed that $\uparrow$ is a reduction function that can be used as a sound strategy pruning method. It is actually the optimal one in terms of cardinality, due to non-locality. Concretely, let c be an arbitrary choice function. If there is $f \in F$ such that $|c(f)| < |f^{\uparrow}|$, then c cannot be used as a sound strategy pruning method for the following reason.

Since $f^{\uparrow} \setminus c(f)$ is not empty, let $p \in f^{\uparrow} \setminus c(f)$. Consider an arbitrary element $q \in f$. If $p \subseteq q$, then $p = q$ by definition of $f^{\uparrow}$. Hence, $|p \cap q| < |q|$ if $p \neq q$. Now consider the vector game in which MIN picks between a leaf with vector p as its

[7] Intuitively, under the preference $\phi_{U'}$, we look for strategies that allow us to win in every world in U'.

reward and a MAX's node, at which MAX picks between leaves leading to vectors in the family f. Then the situational value of the root reads

$$f \sqcap \{p\} = \{p \cap q \mid q \in f\}.$$

In this family, the (only) vector with the largest cardinality is p by the argument above, which means the maxmin value of this game is $|p|$. However, if we apply c to Ginsberg's algorithm, the vector p is pruned from f, hence the vectors in $c(f) \sqcap \{p\}$ have a cardinality strictly smaller than $|p|$. This means applying c yields a wrong maxmin value, therefore c is not a sound strategy pruning method.

As a conclusion, no sound strategy method represented by a choice function can ever prune strictly more strategies than $\uparrow$.

Greedy strategy prunings

We now have a brief look at some very natural but unsound strategy prunings.

We have already seen that keeping only one strategy with maximum support size at each node is not a sound strategy pruning method, again due to non-locality. For example, in the game in Figure 5.6 (page 92), action r does not have maximum support size at node A. However, it is part of the maxmin strategy due to its non-local interaction with action R at node B. If we discard the vector corresponding to l from the situational value $\mathsf{val}(A)$ (and the vector corresponding to L from $\mathsf{val}(B)$), we will not obtain the correct maxmin value because all maxmin strategies are no longer in consideration.

This can be generalised to all $k \in \mathbb{N}$: keeping only k strategies with the largest support size is not sound. Imagine that a universe U with $2n$ different worlds. Let f and g be two families of subsets of U such that

- each family contains $\binom{2n}{n}/2 + 1$ subsets of U;
- each subset of U in f or g has size n;
- only one subset appears in both f and g (i.e. $|f \cap g| = 1$).

Now, imagine a CGII similar to the one in Figure 5.6 (page 92): at r, MIN chooses between node A and node B. At node A, MAX has $\binom{2n}{n}/2 + 1$ different actions, each action leads to a leaf with a different support from the family f. Similarly, at node B actions lead to leaves with different support from the family g. Then $\mathsf{val}(r) = f \sqcap g$, and the maxmin value is n. In addition, the unique maxmin strategy consists in choosing a leaf at node A and a leaf at node B such that these two leaves have the same support; any other strategy has a support of size strictly smaller than n.

Suppose without loss of generality that Ginsberg's algorithm visits node A before node B. To compute the correct maxmin value for this game, a choice function cannot eliminate any strategy from f since we have no information about g yet. For instance, if c eliminates some p from f, then in case $f \cap g = \{p\}$, the algorithm no longer computes the correct maxmin value n. In summary, we must keep all $\binom{2n}{n}/2 + 1$ subsets in f. By varying n, this shows that keeping only k strategies with the largest support is not sound for every fixed k.

On the other hand, although these greedy types of strategy pruning are not sound in general, they are very useful for computing a lower bound of the maxmin value of a game,[8] especially because for all fixed k, keeping only k strategies (in all families

[8] A strategy pruning method is equivalent to restricting MAX's local strategies in a certain way, hence always yields a lower bound for the maxmin value.

computed by the algorithm) limits the combinatorial explosion and leads to a linear running time for Ginsberg's algorithm. In addition, in games with much fewer non-local interactions (unlike the counterexample above), lower bounds computed this way can be very tight, or even exact.[9]

A final note on non-locality

Non-local interactions are computed in our game tree search algorithm by the operator $\sqcap$, which is also the place where all combinatorial explosions arise: a meet of n families of 2 sets can generate a set of size $O(2^n)$. The hope of an efficient implementation of this crucial operator is also shattered by Doyen and Raskin (2011), who show that this operator is NP-hard to compute. This is hardly surprising, since we have already seen evidence of this in the proof of Proposition 5.3.3 (page 76): finding the maximal clique of a graph of size n can be reduced to the computation of the meet of n families of 2 subsets of a set of size n.

In summary, the computational difficulty (notably, the NP-completeness) of computing the maxmin value of a vector game comes from the need to surmount the problem of non-locality, a phenomenon caused by incomplete information. Any pruning or optimisation that does not take non-locality fully into account is doomed to failure, which can be used as a trick to quickly know if a given pruning is not sound.

6.3.3 Maxmin lower bound pruning

Lastly, let us present another sound strategy pruning method. If we know that the maxmin value is bounded from below by n, then we can discard all subsets of cardinality strictly smaller than n. The reason is straightforward: the operators $\cup$ and $\sqcap$ never strictly increase the cardinality of a subset in a family. Indeed, $\cup$ does not change the cardinality of a subset, while $\sqcap$ computes the intersection between this subset and subsets from another family, which results in no larger subsets. Applying this reasoning recursively, we can show that if the maxmin value if at least n, then no local strategy with support size strictly smaller than n can be part of a maxmin strategy; hence it is sound to eliminate the support of such strategies.

For every fixed n, this pruning can be described by as a choice function that can be easily shown to be a reduction function in the sense of Definition 6.2.8 with respect to the weak order induced by $\phi_n(U') = \min(|U'|, n)$ for all $U' \subseteq U$.[10] Hence, for a game with a maxmin value of at least n, we can use this reduction or its approximations (which correspond to a partial elimination of subsets of size strictly smaller than n) as a sound strategy pruning method in Ginsberg's algorithm.

Furthermore, notice that if n is not a lower bound of the maxmin value of a game, this pruning yields a family containing only $\emptyset$ at the root, since every subset in the situational value of the root will be replaced by $\emptyset$. Therefore, maxmin lower bound pruning can be used to decide whether a given threshold of maxmin value is achievable; if it is, then we can even know the exact maxmin value from the output.

Remark. *This seems to contradict the optimality of $\uparrow$, but it actually does not: $\uparrow$ is optimal among all strategy prunings that are sound for* all *games, while this pruning (with the parameter n) is only sound for games with a maxmin value of at least n.*

[9] For a given game, a lower bound computed by using a strategy pruning method is exact if and only if there is a maxmin strategy such that no local strategy involved in it is pruned.

[10] For this, technically we need to assume that all families contain $\emptyset$, and the pruning replaces every subset of size strictly smaller than n by $\emptyset$.

We refer to this kind of pruning as *maxmin lower bound pruning*, to emphasise its reliance on the knowledge of a lower bound. Such a lower bound can be obtained in many ways. For example:

- We can evaluate a randomly drawn strategy of MAX to get a lower bound on the maxmin value.
- We can ameliorate the previous method by a game-specific pre-analysis so that we are capable of drawing better strategies of MAX to get a better lower bound.
- A lower bound is also naturally obtained during the execution of Ginsberg's algorithm when the root is a node of MAX: each time the depth-first algorithm returns to the root, we have a new lower bound provided by the best strategies from the branches already explored.

6.4 Alpha-beta prunings under partial order

We now turn our attention to game tree pruning for Ginsberg's algorithm; we begin with one very generic type of game tree pruning method called alpha-beta pruning. Its applicability is way beyond games with perfect information or CGIIs. The following content in this section has been previously published in IJCAI 2022 (Li et al., 2022).

6.4.1 Related work

Search in graphs containing AND- and OR-nodes is used as a basis of many algorithmic solutions to Artificial Intelligence problems. In such graphs, OR-nodes typically model choice nodes at which an agent can choose a successor, while AND-nodes model an opponent. For instance, in robust planning with nondeterministic actions, an AND-node models the outcome of an action: a strategy must be valid whatever the outcome (Kissmann and Edelkamp, 2009). Similarly, when solving a zero-sum two-player (sequential) game for a player, OR-nodes are those at which it is their turn to play, while AND-nodes correspond to their opponent (van den Herik et al., 2002): the value of an OR-node is 1 if and only if at least one move leads to a node with value 1; dually, at AND-nodes, the values of the children are conjoined. More generally, in games that can have more than two outcomes, such as chess or checkers (Schaeffer et al., 2007), AND-nodes (respectively OR-nodes) correspond to a minimum (respectively maximum) operator on the values of their children.

A fundamental question is that of evaluating rooted AND-OR directed acyclic graphs (DAGs), which means computing the value of their root given a value for each of their leaves. For instance, in games with complete information, selecting the best move for the current turn amounts to evaluating each of the children of the root. This problem has been thoroughly studied in the literature (Marsland, 1986) under the setting of totally ordered values (Boolean or real) and the standard AND/OR or min/max operators. Following Ginsberg and Jaffray (2002), we investigate here a more general setting, where the values are taken from a distributive lattice $(V, \wedge, \vee)$ (i.e. a partially ordered set with a least upper bound and a greatest lower bound for all pairs of elements), and operators for AND- and OR-nodes are taken to be the meet $\wedge$ and join $\vee$, respectively. This setting arises naturally in many applications, in particular in games with incomplete information. Examples of such games are Skat (Kupferschmid and Helmert, 2006; Buro et al., 2009; Rebstock et al., 2019; Edelkamp, 2020), Bridge (Levy,

1989; Ginsberg, 2001; Cazenave and Ventos, 2020), Hearts and Spades (Sturtevant and White, 2006).

A well-known technique for evaluating AND-OR graphs is alpha-beta pruning, which maintains a lower bound α (respectively upper bound β) on the value of each OR-node (respectively AND-node), and uses them to prune some of their successors (Knuth and Moore, 1975; Pearl, 1982; Marsland, 1986). This technique is currently used in strong chess programs (Haworth and Hernandez, 2021) combined with sophisticated evaluation functions such as NNUE neural networks that were first used in Shogi (Nasu, 2018). However, the generalisation of alpha-beta pruning to AND-OR DAGs with partially ordered values is non-trivial since two values from a lattice are not always comparable. We build on the seminal work by Ginsberg and Jaffray (2002) and generalise it by proving the correctness of lattice-valued alpha-beta pruning with the consideration of heuristic functions.

Orthogonally, we investigate caching techniques for alpha-beta pruning in lattice-valued DAGs. The question is again non-trivial because nodes are, in general, revisited with different α and β from during previous visits. For this, we propose a new algorithm called 'alpha-beta duo'. We state its correctness and experimentally evaluate its efficiency.

In the following, we first present the problem setting of AND-OR graph evaluation under partial order and alpha-beta pruning in general. Then we extend the work by Ginsberg and Jaffray (2002) and present alpha-beta duo. Finally, we report experimental results and conclude.

6.4.2 AND-OR graph evaluation under partial order

Throughout this section, we consider *rooted* DAGs, which contain a (necessarily unique) root r such that there exists a directed path to every vertex from r. An *AND-OR DAG* is a rooted DAG the internal nodes of which are partitioned into *AND-nodes* and *OR-nodes*. Note that nodes do not have to be alternating. In games, AND-nodes and OR-nodes are typically considered to be MIN's and MAX's decision nodes, respectively.

Example. *The public tree in a two-player CGII, the internal nodes of which are partitioned into decision nodes for MAX and those for MIN, is an example of an AND-OR DAG that happens to be a tree.*

We are interested in the problem of evaluating the root value of an AND-OR DAG, given values for all its leaves. More concretely, we aim to compute the situational value of the root as defined by the minimax algorithm $\texttt{MiniMax}(V, \mathsf{eval}, \vee, \wedge)$ as in Subsection 5.4.3 (page 93), where $(V, \wedge, \vee)$ forms a bounded distributive lattice.[11] The reason we restrict ourselves to situational values coming from a distributive lattice will be explained in Subsection 6.4.3.

Formally, given an AND-OR DAG G, a bounded distributive lattice $(V, \preceq, \wedge, \vee)$, and an evaluation function $\mathsf{eval} : \mathrm{L}(G) \to V$ assigning a value in V to each leaf of G, the goal is to compute $\mathsf{val}(r)$, where the situational value $\mathsf{val}(n)$ of $n \in N$ is defined recursively by:

- for a leaf node n, $\mathsf{val}(n) \coloneqq \mathsf{eval}(n)$;
- for an internal AND-node n, $\mathsf{val}(n) \coloneqq \bigwedge_{n' \in \mathrm{C}(n)} \mathsf{val}(n')$;

[11] See Appendix A.3.2 for relevant definitions from lattice theory.

- for an internal OR-node n, $\mathsf{val}(n) := \bigvee_{n' \in C(n)} \mathsf{val}(n')$.

Since G is a DAG, the function $\mathsf{val} : N \to V$ is well-defined.

Example. *Consider the DAGs in Figure 6.1, where circle and square nodes represent AND-nodes and OR-nodes, respectively. On the left, the lattice is the set of Boolean vectors of length* 4 *(denoted as words), with bitwise AND and bitwise OR as meet and join, respectively. One can easily verify that* $\mathsf{val}(r) = 1100$*. On the right, the lattice is the set of Boolean vectors of length* 3*, and* $\mathsf{val}(r) = 001$.

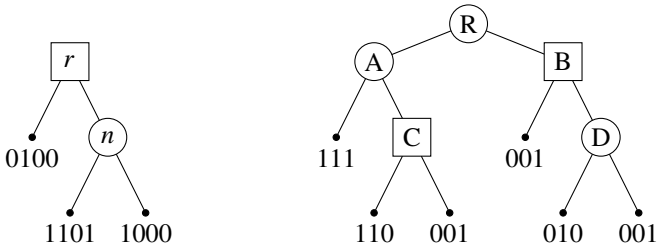


Figure 6.1: Two AND-OR DAGs with partially ordered values.

Example applications

Many important problems are in fact AND-OR DAG evaluation in disguise. The simplest one is Boolean circuit evaluation. Here AND- and OR-nodes model AND and OR gates, the lattice is the Boolean algebra (i.e. $V = \{0, 1\}$ with $0 \preceq 1$ and logical conjunction and disjunction as meet and join), and the evaluation function encodes the inputs of the circuit.

Solving a game with perfect information typically involves computing the maxmin value of a game tree, which can be regarded as evaluating an AND-OR DAG: AND- and OR-nodes are respectively choice nodes of player MIN and of player MAX, the lattice is $(V, \leq, \min, \max)$ with V a totally ordered set such as $\mathbb{Z}$ or $\mathbb{R}$, and the evaluation function gives the rewards for MAX of terminal nodes, or a heuristic value if the search is cut at some depth. Then the situational value of the root is the maxmin value of the game.

In games with incomplete information, non-trivial lattices come into play. For example, we have already seen in Subsection 6.3.1 that Ginsberg's algorithm with reduction $\texttt{MiniMax}(\mathcal{P}(\mathcal{P}(U))^{\uparrow}, \mathsf{eval}, \cup^*, \sqcap^*)$ returns a value of the root from which the maxmin value of the game can be easily deduced. This is another instance of AND-OR DAG evaluation under partial order since Ginsberg (2001) shows that $(\mathcal{P}(\mathcal{P}(U))^{\uparrow}, \cup^*, \sqcap^*)$ forms a bounded distributive lattice.

6.4.3 Alpha-beta pruning

Shallow and deep pruning

Most of the literature on alpha-beta pruning concerns only totally ordered values, such as real numbers. For an overview of alpha-beta pruning in this setting, see the review by Marsland (1986). Since AND-OR DAGs with partially ordered values are useful to

model richer problems, Dasgupta et al. (1996) proposed alpha-beta pruning in this new setting for multi-criteria game.

Ginsberg and Jaffray (2002) gave the first thorough study on this subject; we give here a summary of their results. They consider the setting of AND-OR tree evaluation with a set V and two binary operators $\vee$ and $\wedge$ on V that are associative, commutative, and idempotent.

The notion of shallow pruning is illustrated by the game tree in Figure 6.2 (left). Hence, shallow pruning is sound if and only if $\alpha \vee (v \wedge T) = \alpha$ for all $\alpha, v, T \in V$ with $v \preceq \alpha$, which is actually equivalent to the absorption identity $x \vee (x \wedge y) = x$ for all $x, y \in V$. Therefore, shallow pruning is sound if and only if $(V, \vee, \wedge)$ is a lattice (Ginsberg and Jaffray, 2002, Proposition 3.5).

Similarly, the notion of deep pruning is illustrated by the game tree in Figure 6.2 (right). The soundness of deep pruning is equivalent to the distributivity of $\vee$ and $\wedge$ over each other; hence, deep pruning is sound if and only if $(V, \vee, \wedge)$ is a distributive lattice (Ginsberg and Jaffray, 2002, Proposition 4.2).

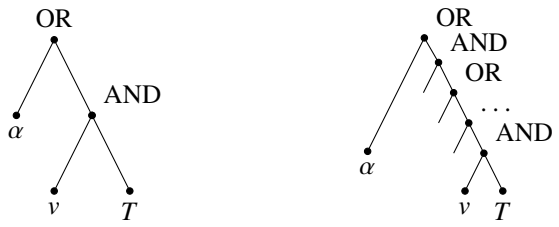


Figure 6.2: Shallow pruning vs. deep pruning; T can be pruned (shallowly or deeply) if $v \preceq \alpha$.

Recall now that Ginsberg and Jaffray (2002) assumed $\vee$ and $\wedge$ to be associative, commutative, and idempotent. These three properties are intrinsically related to assumptions about the rationality of MAX and MIN. Let us take MAX and their operator $\vee$ as an example to illustrate this.

- The associativity of $\vee$ (i.e. $(x \vee y) \vee z = x \vee (y \vee z)$ for all $x, y, z \in V$) is equivalent to saying that the two game trees in Figure 6.3 are equivalent according to MAX.

- Similarly, the commutativity of $\vee$ (i.e. $x \vee y = y \vee x$ for all $x, y \in V$) means MAX is indifferent to the ordering of their actions.

- The idempotency (i.e. $x \vee x = x$ for all $x \in V$) means that for MAX, having two actions with the same results is equivalent to having only one such action.

Hence, to rephrase the result above concerning deep pruning, what Ginsberg and Jaffray (2002) actually showed is that deep pruning is sound for rational players if and only if the set of values is a distributive lattice. However, these properties of rationality do not necessarily hold for human players (Dhami, 2019). Loddo and Saiu (2010) studied the setting in which these properties are violated, and showed that deep pruning is still sound for tropical algebras for some relaxed notion of rationality. For us, it is reasonable enough to assume these properties holds for our setting. Therefore, we focus on the case in which $(V, \vee, \wedge)$ forms a distributive lattice.

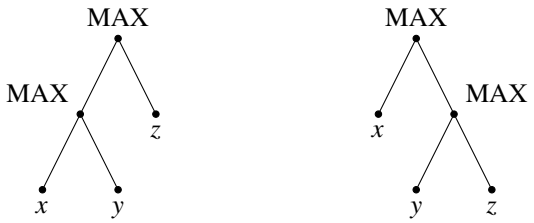


Figure 6.3: Associativity of $\vee$ dictates the equivalence of these two game trees from MAX's perspective.

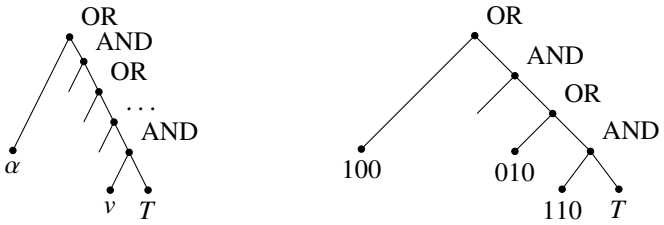


Figure 6.4: Deep pruning vs. expected pruning.

Gaps in the literature

We argue that under partial order, deep pruning, the form of which is given again in Figure 6.4 (left), does not capture every cut an alpha-beta search should perform when values are partially ordered, consider Figure 6.4 (right). The lattice is again the set of Boolean vectors of length 3, with bitwise AND and bitwise OR as meet and join, respectively. When an alpha-beta search algorithm descends to the bottommost AND-node, the value of α would be 110, which is the join of the value of an already explored child of two ancestor OR-nodes. We would like the algorithm to prune the subtree T since the value of its parent node cannot be better than the current value of α (due to the sibling of T). However, deep pruning, as it is defined in the literature (Ginsberg and Jaffray, 2002, for instance), does not apply since the value of α does not come from a child of one single ancestor node. Note that this phenomenon is specific to lattices that are *not* totally ordered, since otherwise the meet or join (i.e. min or max) of two values is always one of them, hence shallow and deep pruning captures all prunings in a standard alpha-beta search.

This loophole of deep pruning can be closed using the insertion identities ((A.1) and (A.2), page 166) of a distributive lattice. For example,

$$x \vee (y \wedge z) = x \vee \big(y \wedge (x \vee z)\big)$$

holds for all $x, y, z \in V$. This means in Figure 6.4 (right), we can insert a value 100 to the leaf with value 010, which yields $100 \vee 010 = 110$ and triggers the pruning of T.

However, there is also another question not formally addressed in the literature, which is the initialisation of node values. In standard alpha-beta search, one typically initialises the value of an OR-node (respectively AND-node) to be α (respectively β) (Knuth and Moore, 1975) or $-\infty$ (respectively $+\infty$) (Marsland, 1986) (note that $-\infty$ and $+\infty$ translate to $\bot$ and $\top$ in our context). However, one may have access to a heuristic evaluation of nodes, typically by evaluating a relaxed version of the problem which

is easier to solve. For instance, a player cannot do better in a game with incomplete information than in the same game but with complete information. The latter being much easier to solve, the value obtained can be used as a heuristic in the original game with incomplete information. Ideally, initialising values with an accurate heuristic should accelerate the search by finding cuts earlier.

Therefore, in the next subsection, we will give a formal treatment to this subject and formally prove, once for all, that the expected pruning in Figure 6.4 (right) is indeed sound, even when combined with a heuristic function, provided that the heuristic function we use satisfies some admissibility.

6.4.4 Alpha-beta search with heuristic function

Algorithm 4: Alpha-beta search

```
1  def AlphaBeta(node n, α, β):
2      I = h(n, α, β)
3      v ← I
4      determine the successor nodes n_1, ..., n_b of n
5      for i in {1, ..., b}:
6          if n is an OR-node:
7              α ← α ∨ v
8          else:
9              β ← β ∧ v
10         if α ⪰ β:
11             break
12         v_child ← AlphaBeta(n_i, α, β)
13         if n is an OR-node:
14             v ← v ∨ v_child
15         else:
16             v ← v ∧ v_child
17     return v
```

To fill these two gaps in the literature, we first formalise alpha-beta search under partial order with initialisation function in Algorithm 4. We denote by h the initialisation function. In general, its value depends on the current α and β, so we define it to yield a value $h(n, \alpha, \beta) \in V$ for all nodes n and $\alpha, \beta \in V$. Note that since a non-trivial initial value can be used for a node n (Line 3), a cut may happen even before the first child of n is explored; hence we update α and β (Line 7 and Line 9) and determine whether there is a cut (Line 10) at the beginning of the main loop. This is otherwise the same algorithm as in the literature (Marsland, 1986, for instance).

It can be seen that Algorithm 4 will perform the wished pruning in the example in Figure 6.4 (right). By mimicking the proof by Knuth and Moore (1975), we prove that such pruning is indeed sound, thereby extending the result by Ginsberg and Jaffray (2002) to its full form, provided that the initialisation function h satisfies a certain admissibility condition.

Definition 6.4.1 (Admissible heuristic function)**.** *A heuristic function h is said to be*

admissible *for G and V if for all nodes n in G and all $\alpha, \beta \in V$,*

$$h(n,\alpha,\beta)\begin{cases} = \mathsf{val}(n) & \text{if } n \text{ is a leaf node;} \\ \succeq \mathsf{val}(n) \wedge \beta & \text{if } n \text{ is an AND-node;} \\ \preceq \mathsf{val}(n) \vee \alpha & \text{if } n \text{ is an OR-node.} \end{cases}$$

Note that this condition is satisfied by the initialisation values usually used in the literature, such as α or $-\infty$ for OR-nodes (and β or $+\infty$ for AND-nodes). It is also satisfied when $h(n,\alpha,\beta)$ overestimates $\mathsf{val}(n)$ for internal AND-nodes and underestimates it for internal OR-nodes.

We denote the value returned by Algorithm 4 with input n, α, β by $f(n,\alpha,\beta)$. The correctness of Algorithm 4 is a consequence of the following central result.

Proposition 6.4.2. *If h is an admissible heuristic function for G and V, then for all nodes n of G and all $\alpha, \beta \in V$, we have*

$$\beta \wedge \big(\alpha \vee f(n,\alpha,\beta)\big) = \beta \wedge \big(\alpha \vee \mathsf{val}(n)\big). \tag{6.9}$$

Proof idea. This is proved by a structural induction on n and an induction to relate the value of v after each loop to the value returned for the children of n. See the appendix for the full proof. □

Intuitively, Proposition 6.4.2 states that the value returned by Algorithm 4 is the exact situational value of n up to a factor of α and β. Hence, even for partially ordered values, alpha-beta search can be interpreted as search with a pruning window.

Now it follows that Algorithm 4 is correct in the sense that if we use a lower and an upper bound as the initial search window at a node, we can recover its exact situational value using the value returned by the algorithm.

Corollary 6.4.3. *If h is admissible for G and V, then for all nodes n of G and all $\underline{v}, \overline{v} \in V$ satisfying $\underline{v} \preceq \mathsf{val}(n) \preceq \overline{v}$, we have $\mathsf{val}(n) = \overline{v} \wedge \big(\underline{v} \vee f(n,\underline{v},\overline{v})\big)$. In particular, $\mathsf{val}(n) = f(n,\bot,\top)$.*

Proof. Plug $\alpha = \underline{v}$ and $\beta = \overline{v}$ into (6.9), and use the fact that $\underline{v} \vee \mathsf{val}(n) = \mathsf{val}(n) = \overline{v} \wedge \mathsf{val}(n)$. □

Importantly, contrary to the case of totally ordered values, generally it is *not* true that the stronger equality $\mathsf{val}(n) = f(n,\underline{v},\overline{v})$ holds, as the example in Figure 6.1 (left) shows. Let $h(r,\cdot,\cdot) = \bot = 0000$ and $h(n,\cdot,\cdot) = \top = 1111$, so that h is admissible. Recall that $\mathsf{val}(r) = 1100$. For $\underline{v} = 1000$ and $\overline{v} = 1110$, indeed $\underline{v} \preceq \mathsf{val}(r) \preceq \overline{v}$. However, $f(r,\underline{v},\overline{v}) = 1101 \neq \mathsf{val}(r)$ since an α-cut happens in the call on n (Line 10) after examining its first child, since at this point $\alpha = \beta = 1100$. However, since $\alpha = \underline{v}$ and $\beta = \overline{v}$ yield no constraint on the fourth component of the values, we still have $\mathsf{val}(r) = \overline{v} \wedge \big(\underline{v} \vee f(r,\underline{v},\overline{v})\big)$, as stated in Corollary 6.4.3.

6.4.5 Alpha-beta duo algorithm

The trouble of standard alpha-beta search is that the returned value is equal to the exact situational value only up to a factor of α and β. It is therefore a non-trivial question how to reuse a previously computed value, since subsequent revisits of a node may come with different search windows.

In alpha-beta search with cache for totally ordered values (Marsland, 1986, for instance), one can exploit the fact that with usual value initialisation, the value $f(n, \alpha, \beta)$ satisfies $f(n, \alpha, \beta) < \beta \Rightarrow \mathsf{val}(n) \leq f(n, \alpha, \beta)$ and $f(n, \alpha, \beta) > \alpha \Rightarrow \mathsf{val}(n) \geq f(n, \alpha, \beta)$. In particular, if $\alpha < f(n, \alpha, \beta) < \beta$, then $f(n, \alpha, \beta) = \mathsf{val}(n)$. Hence, by comparing $f(n, \alpha, \beta)$ to α and β, one can determine whether it is exact or a lower/upper bound, and store it in the cache with an appropriate flag.

However, this does not hold in general for partially ordered values, as shown on Figure 6.1 (right). For $\alpha = 010$ and $\beta = 110$, a β-cut happens after evaluating the first child of C, and an α-cut after evaluating the first child of D. Hence, the algorithm returns 010 for R, which is neither a lower nor an upper bound of the exact situational value 001. In fact, these two values are incomparable in the lattice. If R is an internal node in a DAG, then caching this returned value 010 may cause an evaluation error when the algorithm revisits R.

To tackle this difficulty, we propose a new algorithm named 'alpha-beta duo', which computes a pair of values for all nodes instead of one single value.[12] The algorithm is presented in Algorithm 5, where Cache refers to a transposition table the entries of which are pairs of values indexed by nodes of G, and $(\underline{h}, \overline{h})$ refers to a pair of initialisation functions for which we assume the following property.

Definition 6.4.4 (Admissible initialisation function)**.** *A pair $(\underline{h}, \overline{h})$ of initialisation functions is said to be* admissible *for G and V if for all nodes n in G, $\underline{h}(n) \preceq \mathsf{val}(n) \preceq \overline{h}(n)$ holds, and in addition, if n is a leaf node, $\underline{h}(n) = \overline{h}(n) = \mathsf{val}(n)$ holds.*

In other words, admissible $\underline{h}$ and $\overline{h}$ respectively underestimates and overestimates the value of a node. Note that $\underline{h}$ and $\overline{h}$ that assign respectively $\bot$ and $\top$ to all internal nodes form an admissible pair, which can always be used if one does not have better heuristic functions.

Alpha-beta duo search works in the following manner.

- First, variables $\underline{c}$ and $\overline{c}$ denote respectively the best lower and upper bound of $\mathsf{val}(n)$ available to the algorithm before this call. If n has already been visited, then $\underline{c}$ and $\overline{c}$ are retrieved from the cache. Otherwise, they are given by the initialisation functions $\underline{h}$ and $\overline{h}$. If $\underline{c} = \overline{c}$, $(\underline{c}, \overline{c})$ is returned immediately.

- Otherwise, by symmetry, consider the case in which n is an OR-node. During the main loop, $\underline{v}$ and $\overline{v}$ are respectively the cumulative lower and upper bound of $\mathsf{val}(n)$ (notice that they are both initialised to $\bot$ for an OR-node). If a cut ever happens, it means not all children of n have been evaluated, hence $\overline{v}$ is not a valid upper bound of $\mathsf{val}(n)$. Then we take $\overline{v}$ to be $\overline{c}$, the best upper bound previously known. On the other hand, $\underline{v}$, which is the join of lower bounds of children of n that have been evaluated, is a valid lower bound, so we keep it.

- Finally, after the main loop, $\underline{v}$ and $\overline{v}$ are the lower and upper bounds of $\mathsf{val}(n)$ computed by the current call. Hence, they are combined with the previously known bounds $\underline{c}$ and $\overline{c}$ to yield to the best currently known bounds on $\mathsf{val}(n)$ and they are cached.

We now prove that the alpha-beta duo algorithm is correct. For this, we first need the following notion.

[12] Retrospectively, we found out that a similar idea has been studied by Atzmon et al. (2018) to compute an approximation (within a given error bound) of the maxmin value of a game with perfect information.

Algorithm 5: Alpha-beta duo search

```
1  def AlphaBetaDuo(node n, α, β):
2      if there is an entry for n in the cache:
3          (c̲, c̄) ← Cache(n)
4      else:
5          (c̲, c̄) ← (h̲(n), h̄(n))
6      if c̲ = c̄:
7          return (c̲, c̄)
8      α ← α ∨ c̲
9      β ← β ∧ c̄
10     if n is an OR-node:
11         (v̲, v̄) ← (⊥, ⊥)
12     else:
13         (v̲, v̄) ← (⊤, ⊤)
14     determine the children n_1, ..., n_b of n
15     for i in {1, ..., b}:
16         if α ⪰ β:
17             if n is an OR-node:
18                 v̄ = c̄
19             else:
20                 v̲ = c̲
21             break
22         v̲′, v̄′ ← AlphaBetaDuo(n_i, α, β)
23         if n is an OR-node:
24             v̲ ← v̲ ∨ v̲′
25             v̄ ← v̄ ∨ v̄′
26             α ← α ∨ v̲′
27         else:
28             v̲ ← v̲ ∧ v̲′
29             v̄ ← v̄ ∧ v̄′
30             β ← β ∧ v̄′
31     v̲ ← v̲ ∨ c̲
32     v̄ ← v̄ ∧ c̄
33     store (v̲, v̄) in the cache under an entry for n
34     return (v̲, v̄)
```

Definition 6.4.5 (Coherent cache)**.** *A cache* Cache *is said to be* coherent *for G and V if for all nodes n in G, if there is an entry for n in the cache, then* $\mathsf{Cache}(n) = (\underline{c}, \overline{c})$ *where* $\underline{c} \preceq \mathsf{val}(n) \preceq \overline{c}$*, and in addition, if n is a leaf node, then* $\underline{c}(n) = \overline{c}(n) = \mathsf{val}(n)$.

Obviously, an empty cache is coherent for all G and V.

In the following, we denote the pair of values returned by Algorithm 5 with input n, α, β by $(\underline{f}(n, \alpha, \beta), \overline{f}(n, \alpha, \beta))$. We first show that if the cache is initially coherent, then it remains coherent after the execution, and that no interval stored in it can become looser.

Proposition 6.4.6. *If* $(\underline{h}, \overline{h})$ *is admissible and* Cache *is initially coherent for G and V,*

then for all nodes n in G and all $\alpha, \beta \in V$, we have

$$\underline{f}(n, \alpha, \beta) \preceq \mathsf{val}(n) \preceq \overline{f}(n, \alpha, \beta). \tag{6.10}$$

Moreover, if there is an entry $(\underline{c}, \overline{c})$ in the cache for n before the call, then $\underline{c} \preceq \underline{f}(n, \alpha, \beta)$ and $\overline{f}(n, \alpha, \beta) \preceq \overline{c}$.

Proof idea. $\underline{c} \preceq \underline{f}(n, \alpha, \beta)$ and $\overline{f}(n, \alpha, \beta) \preceq \overline{c}$ are direct consequence of Line 31 and Line 32. As for inequality (6.10), it can be established by a structural induction. See the appendix for the full proof. □

We can now prove results parallel to those in Subsection 6.4.3.

Proposition 6.4.7. *If $(\underline{h}, \overline{h})$ is admissible and* Cache *is initially coherent for G and V, then for all nodes n in G and all $\alpha, \beta \in V$ we have*

$$\beta \wedge \big(\alpha \vee \underline{f}(n, \alpha, \beta)\big) = \beta \wedge \big(\alpha \vee \overline{f}(n, \alpha, \beta)\big). \tag{6.11}$$

Proof idea. This is also proved by a structural induction on n, with the help of Proposition 6.4.6. See the appendix for the full proof. □

Corollary 6.4.8. *If $(\underline{h}, \overline{h})$ is admissible and* Cache *is initially coherent for G and V, then for all nodes n in G and all $\underline{v}, \overline{v} \in V$ satisfying $\underline{v} \preceq \mathsf{val}(n) \preceq \overline{v}$, we have $\mathsf{val}(n) = \overline{v} \wedge \big(\underline{v} \vee \underline{f}(n, \underline{v}, \overline{v})\big) = \overline{v} \wedge \big(\underline{v} \vee \overline{f}(n, \underline{v}, \overline{v})\big)$. In particular, $\mathsf{val}(n) = \underline{f}(n, \bot, \top) = \overline{f}(n, \bot, \top)$.*

Proof. By Proposition 6.4.6, $\underline{f}(n, \underline{v}, \overline{v}) \preceq \mathsf{val}(n) \preceq \overline{f}(n, \underline{v}, \overline{v})$. Using $\overline{v} \wedge \big(\underline{v} \vee \mathsf{val}(n)\big) = \mathsf{val}(n)$, we get

$$\overline{v} \wedge \big(\underline{v} \vee \underline{f}(n, \underline{v}, \overline{v})\big) \preceq \mathsf{val}(n) \preceq \overline{v} \wedge \big(\underline{v} \vee \overline{f}(n, \underline{v}, \overline{v})\big).$$

These inequalities are in fact equalities, since from Proposition 6.4.7 we have $\overline{v} \wedge \big(\underline{v} \vee \underline{f}(n, \underline{v}, \overline{v})\big) = \overline{v} \wedge \big(\underline{v} \vee \overline{f}(n, \underline{v}, \overline{v})\big)$. □

6.4.6 Experiments

To assess the efficiency of alpha-beta duo (hereafter 'ABD'), we ran experiments comparing it to three other algorithms:

- alpha-beta without cache (Algorithm 4, 'AB' for short);
- an alpha-beta search that only caches the value computed for a node if no cut is found during the search in the subtree rooted at this node (hereafter 'ABC');
- a minimax search algorithm without alpha-beta pruning, but with a cache (hereafter 'MMC').

The code of ABD was slightly optimized by refining the values computed on Line 31 and Line 32 with the corresponding bounds of all fully explored children. It is easy to show that this preserves the correctness of the algorithm.

For all experiments, we measured the number of nodes of the DAG visited at least once, the total number of node visits (equivalently, the total number of recursive calls), and the time taken for solving the problem. Intuitively, we expect ABD to be better than

ABC, ABC to be better than MMC (because MMC does not use alpha-beta pruning), and MMC to be better than AB (because the latter does not cache its results and hence, recomputes several times for the same node).

We used two synthetic sets of benchmarks. The first (hereafter 'random') consists of random DAGs of the same kind as the one in Figure 6.1. Random DAGs with parameters d (depth), b (branching factor), and v (number of variables) are generated in the following manner:

- d layers $0, 1, \ldots, d-1$ are built: layer i consists of $3^{\min(i, \frac{3d}{2} - i)}$ nodes (which yields diamond-shapes DAGs);
- from each node n, a set of b nodes is randomly chosen from nodes in the next layer to be $\mathrm{C}(n)$;
- each internal node is labelled AND or OR at random;
- the value of each terminal node is a uniformly drawn subset of $\{1, \ldots, v\}$, or equivalently a random Boolean vector of length v.

We also consider strictly alternating DAGs, in which all nodes in layer i are OR-nodes (respectively AND-nodes) if i is even (respectively odd). In particular, the root is an OR-node.

The second set of benchmarks consists of a simplified version of the card game Bridge that we call 'racing'. There are two players, MIN and MAX. Each has a hand of h cards drawn uniformly from the deck $\{1, \ldots, d\}$ where $d \geq 2h$. Players only see their own hand. MIN begins the game. During each trick, the player who begins plays a card from their hand, the other sees it and plays a card in turn. The player who played the highest card wins this trick and starts the next one. No new card is ever drawn from the deck. The game ends when the players have no more cards or when one has won in total g tricks. MAX wins if they are the first one to reach g tricks. For the benchmark, each instance with parameter h, d, and g consists of a randomly drawn hand with h cards for MAX and a randomly drawn card that is supposed to be played by MIN during the first trick. Notice that when $d > 2h$, each player has incomplete information. We use evaluation of AND-OR DAGs to compute optimal strategies for MAX against the best-defence model defined by Frank and Basin (1998) (cf. Subsection 5.4.1).

In games with incomplete information, in which S is the set of all possible hidden configurations, Ginsberg's algorithm with reduction makes use of the bounded distributive lattice $(\mathcal{P}(\mathcal{P}(S))^{\uparrow}, \cup^*, \sqcap^*)$ (cf. Subsection 6.3.1) to compute uniform strategies. One can also be interested in non-uniform strategies, which allow player MAX's actions to depend on the hidden information. This can be useful for computing heuristic values of for the game with incomplete information. It can be seen that the set of all configurations for which there is a non-uniform winning strategy can be computed as the value of the game DAG with the lattice $(\mathcal{P}(S), \subseteq, \cap, \cup)$ (cf. Subsection 6.5.2).

Hence, for both benchmarks, we consider the two lattices $(\mathcal{P}(\mathcal{P}(S))^{\uparrow}, \cup^*, \sqcap^*)$ and $(\mathcal{P}(S), \cap, \cup)$. In 'random', $S = \{1, \ldots, v\}$, while in 'racing' S is the set of all possible hands of player MIN.

For space reasons, we only give the most representative results, in terms of computation time. For each parameter setting and each algorithm, we averaged over 10 runs. Figure 6.5 shows two examples in which, as can be expected, it is more efficient to cache bounds, even more to perform cuts, and still more to compute and store two bounds per node. On the top plot, the gain of using ABD is exponential: with the branching factor

increasing, ABD gets a better advantage of computing and caching two bounds. On the bottom plot, ABC and AB (not represented) are exponentially worse, and ABD is better than MMC when the branching factor is high.

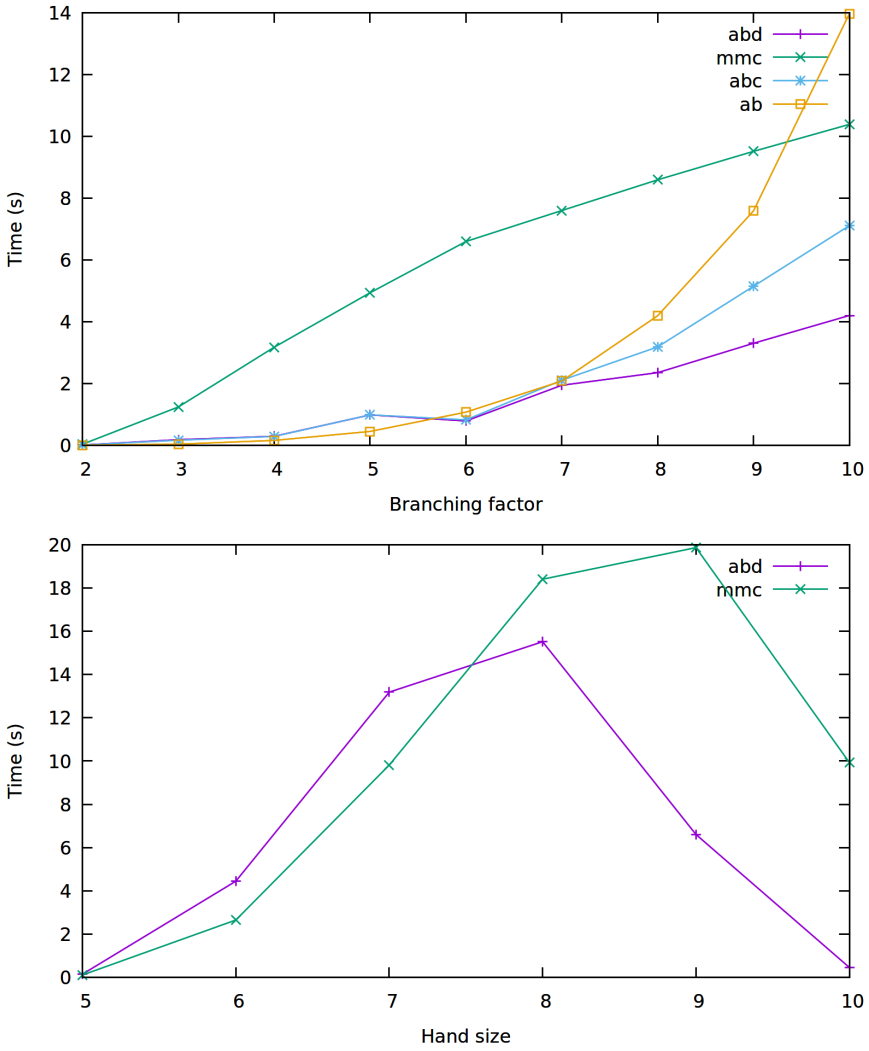


Figure 6.5: Experimental results on random (top) and racing (bottom). Top: $d = 15$, $v = 10$ (varying b), alternating DAGs. Bottom: $d = 20$, $g = 5$ (varying h). The lattice is $\mathcal{P}(S)$ in both cases.

Now Figure 6.6 shows two examples in which it turns out that it is not always better to use alpha-beta pruning with cache.

On the top plot, not caching results at all turns out to be better: the overhead due to the additional operators from the lattice $\mathcal{P}(\mathcal{P}(S))^{\uparrow}$ (which are necessary to maintain the cache) seems to compensate for the advantage of ABC or ABD in terms of number of visited nodes and recursive calls (the curves are reversed for this metrics, not shown here). On the bottom plot, it turns out that sometimes alpha-beta pruning even degrades

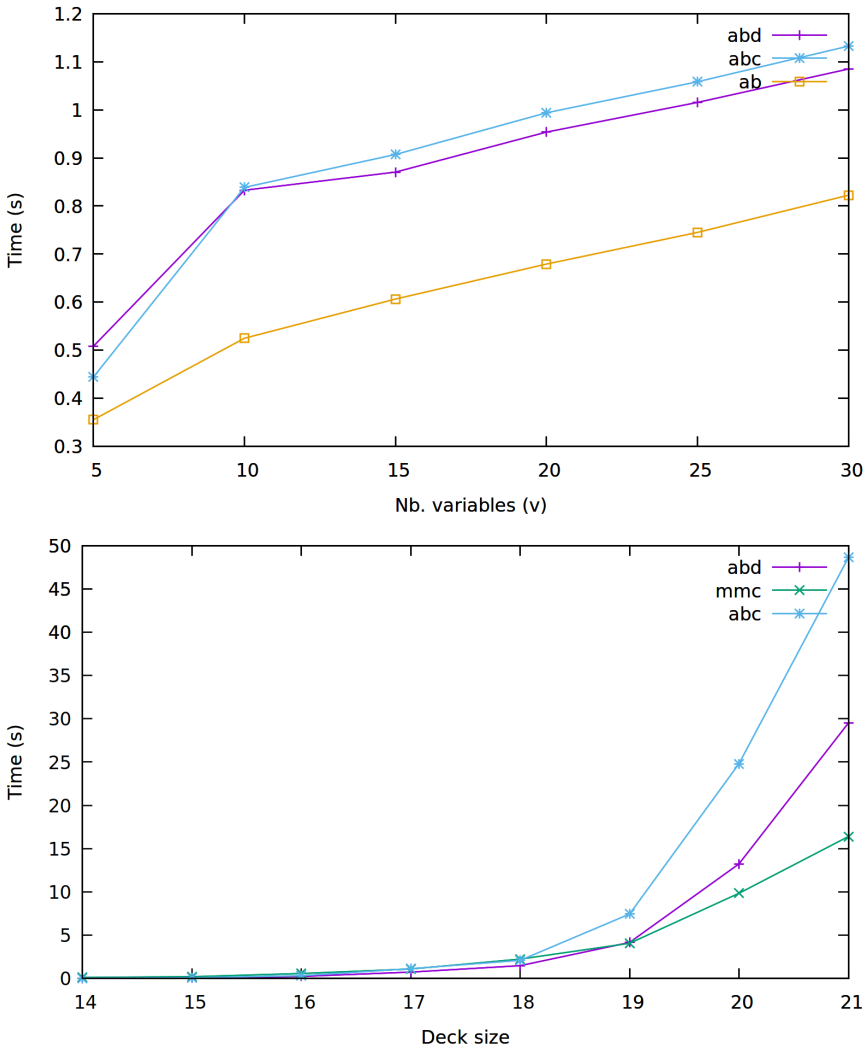


Figure 6.6: Experimental results on random (top) and racing (bottom). Top: $d = 15$, $b = 4$ (varying v), alternating DAGs, lattice $\mathcal{P}(\mathcal{P}(S))^{\uparrow}$. Bottom: $h = 7$, $g = 5$ (varying d), lattice $\mathcal{P}(S)$.

performance. Again, this is due to the overhead of manipulating values from a large lattice (for a fixed hand size, the lattice grows exponentially with the deck size).

To complete the discussions on these results, let us mention that in most experiments, the numbers of nodes explored and visited by each method are ordered as expected, with ratios varying from linear to exponential. In particular, for these metrics, ABD is most of the time better, and often much better, than the other three algorithms.

Summarising, ABD seems to provide a real gain in (brute) performance for DAGs with high branching factors. Contrastingly, when $\wedge$ and $\vee$ from the lattice are too expensive to compute (as is the case in some large lattices), it may sometimes be better not to use cache and alpha-beta pruning together, due to the overhead to maintain the

coherence of the cache.

6.4.7 Conclusion

We investigated alpha-beta search for AND-OR DAGs with values from a lattice, which has direct applications such as solving games with incomplete information. We have extended previous formal analyses, in particular to the use of heuristic as initialisation functions. Then we have proposed a new algorithm named 'alpha-beta duo', which caches both a lower and an upper bound of the value of each visited node, and we have formally proved its correctness. Experiments strongly suggest that it is more efficient than other algorithms in terms of number of visited nodes and recursive calls. As for time efficiency, the alpha-beta duo algorithm turns out to be more efficient than other algorithms for DAGs with large branching factors and lattices of reasonable size. As an interesting conclusion, our experiments also put forth that in other cases, it may be better not to use alpha-beta pruning at all, due to the additional overhead of maintaining a correctly reusable cache.

6.5 Other types of game tree pruning

In the last section, we have presented how to apply alpha-beta pruning to AND-OR graph with partially ordered value. And we have observed that the value returned for a node is the situational value of the node up to a factor of α and β. From another perspective, α and β serve as a search window and an alpha-beta search only computes a *masked* or *projected* situational value with respect to the window.

Occasionally, we may have information that can be used as an additional search window to let the search focus only on the essential part of the situational values. However, the value of α and β at each node are mechanically determined by Algorithm 4 and do not have the place to incorporate this sort of information. To remediate this, we introduce a third variable γ to represent this additional search window or mask on the situational values. In this section, we will first present this extension of alpha-beta search that we call *alpha-beta-gamma search*. We then discuss some instantiations of this algorithm as optimisations for Ginsberg's algorithm.

6.5.1 Alpha-beta-gamma search algorithm

Alpha-beta-gamma search under partial order is formalised in Algorithm 6. The only difference between Algorithm 6 and Algorithm 4 is the addition of the variable γ, the value of which is in V and which is used to mask the initial value of a node, and the *gamma update function* g. The initial value of γ and the gamma update function can come from human advice (e.g. when a human expert wishes to guide the search to a particular window they deem more pertinent), or some pre-analyses on the game tree (cf. Subsection 6.5.2 and Subsection 6.5.4).

Remark. *In Algorithm 6, we allow the gamma update function g to depend solely on the node n at which we carry out the update, and on the current value of γ. Naturally, this is not the most general form of update; one can, for example, allow g to also depend on the current value of α and β. However, as the dependency on n and γ is sufficient to formulate all game tree prunings we will present in the following, we adopt this more restricted version of the gamma update function in Algorithm 6, and leave a more general version of this function for future work.*

Algorithm 6: Alpha-beta-gamma search

```
1  def AlphaBetaGamma(node n, α, β, γ):
2      γ' ← g(n, γ)
3      I = h(n, α, β)
4      v ← γ' ∧ I
5      determine the successor nodes n_1, ..., n_b of n
6      for i in {1, ..., b}:
7          if n is an OR-node:
8              α ← α ∨ v
9          else:
10             β ← β ∧ v
11         if α ⪰ β:
12             break
13         v_child ← AlphaBetaGamma(n_i, α, β, γ')
14         if n is an OR-node:
15             v ← v ∨ v_child
16         else:
17             v ← v ∧ v_child
18     return v
```

Definition 6.5.1 (Admissible gamma update function)**.** *A gamma update function g is said to be* node-admissible *for G and V if for all nodes n of G, it holds that*

$$\forall \gamma \in V, g(n, \gamma) \succeq \mathsf{val}(n).$$

g is said to be path-admissible *for G and V if for all nodes n of G, it holds that*

$$\forall \gamma \in V, \gamma \succeq g(n, \gamma) \succeq \gamma \wedge \mathsf{val}(n).$$

g is said to be admissible *if it is either node-admissible or path-admissible.*

Remark. *The second notion of admissibility is called* path-*admissibility because it allows $g(n, \gamma)$, the updated value of γ at node n, to depend on the current value of γ, which itself, due to the recursive nature of the algorithm, depends on the value of γ along the path which leads to the current node n. On the other hand, purely local update functions, which depend solely on the current node but not on the current value of γ, can be cast into the* node-*admissible category.*

This definition of admissibility is both motivated by game tree prunings we will present in the following and by the proof of correctness of Algorithm 6. However, it is only a sufficient admissibility for Algorithm 6 to be correct, not a necessary one. See the remark after the proof of Proposition 6.5.2, which we state below, in the appendix.

We denote the value returned by Algorithm 6 with input n, α, β, γ by $f(n, \alpha, \beta, \gamma)$. The correctness of Algorithm 6 will be a direct consequence of the following central result, the proof of which is based on structural induction on n and can be found in the appendix:

Proposition 6.5.2. *If h is an admissible heuristic function and g is an admissible gamma update function for G and V, then for all nodes n of G and all $\alpha, \beta, \gamma \in V$, we*

have

$$\beta \wedge \big(\alpha \vee f(n,\alpha,\beta,\gamma)\big) = \beta \wedge \Big(\alpha \vee \big(g(n,\gamma) \wedge \mathsf{val}(n)\big)\Big), \tag{6.12}$$

$$\alpha \vee \big(\beta \wedge f(n,\alpha,\beta,\gamma)\big) = \alpha \vee \Big(\beta \wedge \big(g(n,\gamma) \wedge \mathsf{val}(n)\big)\Big). \tag{6.13}$$

Remark. *Notice that Proposition 6.5.2 is a generalisation of Proposition 6.4.2. Indeed, taking $g(n,\gamma) = \top$ for all n and γ, which is a node-admissible gamma update function, Algorithm 6 reduces to Algorithm 4, while* (6.12) *reduces to* (6.9).

Proposition 6.5.2 also highlights how gamma (and the gamma update function) and the initialisation function differ in their effect: an admissible initialisation function does not have any consequence on the returned values, while gamma projects the returned value onto a window defined by the gamma update function.

Intuitively, Proposition 6.5.2 states that the value returned by Algorithm 6 coincides with the exact situational value of n masked by γ' (i.e. $g(n,\gamma) \wedge \mathsf{val}(n)$) up to a factor of α and β. Now it follows that Algorithm 6 is correct in the following sense.

Corollary 6.5.3. *If h and g are admissible for G and V, then for all nodes n of G, we have*

- *if g is node-admissible,* $\mathsf{val}(n) = f(n,\bot,\top,\gamma)$*;*
- *if g is path-admissible,* $\gamma \wedge \mathsf{val}(n) = f(n,\bot,\top,\gamma)$.

Proof. For $\gamma \in V$, and all $\underline{v}_\gamma, \overline{v}_\gamma \in V$ satisfying $\underline{v}_\gamma \preceq g(n,\gamma) \wedge \mathsf{val}(n) \preceq \overline{v}_\gamma$, by plugging $\alpha = \underline{v}_\gamma$ and $\beta = \overline{v}_\gamma$ into (6.12) and (6.13) we have

$$g(n,\gamma) \wedge \mathsf{val}(n) = \overline{v}_\gamma \wedge \big(\underline{v}_\gamma \vee f(n,\underline{v}_\gamma,\overline{v}_\gamma,\gamma)\big) = \underline{v}_\gamma \vee \big(\overline{v}_\gamma \wedge f(n,\underline{v}_\gamma,\overline{v}_\gamma,\gamma)\big).$$

In particular, when $\underline{v}_\gamma = \bot$ and $\overline{v}_\gamma = \top$ we have

$$g(n,\gamma) \wedge \mathsf{val}(n) = f(n,\bot,\top,\gamma).$$

If g is node-admissible, then $g(n,\gamma) \wedge \mathsf{val}(n) = \mathsf{val}(n)$; if g is path-admissible, then

$$\gamma \wedge \mathsf{val}(n) \succeq g(n,\gamma) \wedge \mathsf{val}(n) \succeq \gamma \wedge \mathsf{val}(n) \wedge \mathsf{val}(n),$$

hence $g(n,\gamma) \wedge \mathsf{val}(n) = \gamma \wedge \mathsf{val}(n)$. □

Remark. *Just as Proposition 6.5.2 generalises Proposition 6.4.2, Corollary 6.5.3 implies Corollary 6.4.3.*

6.5.2 Prunings with winning supports under complete information

We now study a few instances of alpha-beta-gamma prunings that are inspired by human players' search and planning techniques in the gameplay of Bridge.

Until now, we have treated the situational values of internal nodes as abstract elements from a bounded distributive lattice. However, they can contain exploitable information if we look at them at a lower granularity.

In the following, we denote by $\mathsf{val}(n)$ the situational value of node n computed by Ginsberg's algorithm with reduction (i.e. $\texttt{MiniMax}(\mathcal{P}(\mathcal{P}(U)), \mathsf{eval}, \cup^*, \sqcap^*)$ with $\mathsf{eval}(l) = \{\|\vec{u}(l)\|\}$[13] for all leaf nodes l) for a Boolean vector game. Recall that $\mathsf{val}(n)$ is a family of the universe U, and each subset of U in it is the support of a non-dominated local strategy of MAX at n.

[13] Recall that $\|\vec{v}\| \in \mathcal{P}(U)$ is the subset of U corresponding to the binary vector $\vec{v} \in \mathbb{B}^t$.

Complete information vectors as gamma

Before introducing Ginsberg's algorithm, we have presented a failed attempt to modify the minimax algorithm, which can be written as $\texttt{MiniMax}(\mathcal{P}(U), \mathsf{eval}, \cup, \cap)$, where $\mathsf{eval}(l) = \|\vec{u}(l)\|$ for all leaf nodes $l \in \mathrm{L}(T)$. We will refer to this algorithm as the *CI algorithm*.[14] The CI algorithm does not correctly compute the pure maxmin of a vector game due to strategy fusion. However, it does compute valuable information that we can exploit for prunings.

Let us denote by $\mathrm{CI}(n)$ the situational value of $n \in \mathrm{N}(T)$ computed by the CI algorithm. Notice that this value, which is a subset of U, is computed by set union and intersection (or equivalently, bit-wise maximum and minimum for binary vectors). Hence, $\mathrm{CI}(n)$ is easily interpretable: if $\omega \in \mathrm{CI}(n)$, this means MAX could force a win in this world starting at node n, *if* they had complete information.[15] On the other hand, if $\omega \notin \mathrm{CI}(n)$, this means MIN can force a win (i.e. a loss for MAX) in this world starting at node n.

In the following, we refer to a strategy with which a MAX with complete information could achieve the value $\mathrm{CI}(n)$ as MAX's *optimal local strategy under complete information* at n. We also refer to $\mathrm{CI}(n)$ as MAX's *winning support under complete information* at n and the vector corresponding to this set as the *CI vector* at n.

Take a look again at the vector game in Figure 5.9 (page 95). We have seen that $\mathrm{CI}(r) = (1, 1, 0, 1, 1)$. It can be easily verified that under complete information, MAX could indeed force a win in all worlds except the third one, by choosing the appropriate action between l and r as a function of the real world. More concretely, MAX should play l in the first two worlds and r in the last two. Notice that this optimal strategy under complete information is actually not a legal strategy of MAX in the vector game in Figure 5.9, in which MAX does have incomplete information and is forced to play the same action in all 5 worlds. This explains the discrepancy between the *CI vector* at r, which indicates a pure maxmin value of $4/5$ if MAX had complete information, and the situational value $\mathsf{val}(r)$, which shows that the pure maxmin value of the vector game is only $2/5$.

It is straightforward to show the following result by induction.

Lemma 6.5.4. *Let n be a node and $U' \in \mathsf{val}(n)$ be a support. Then $U' \subseteq \mathrm{CI}(n)$.*

Remark. *This intuitively means no local strategy of MAX at n has larger support than MAX's optimal strategy under complete information at the same node.*

Hence, the family $\{\mathrm{CI}(n)\}$ is an upper bound of $\mathsf{val}(n)$. In other words, $\mathsf{val}(n) \preceq \{\mathrm{CI}(n)\}$, where $\preceq$ is the partial order from the lattice of reduced families used in Ginsberg's algorithm with reduction (cf. Subsection 6.3.1).

As a result, the gamma update function $g(n, \gamma) = \{\mathrm{CI}(n)\}$ is node-admissible, therefore by Corollary 6.5.3, for all nodes n and all $\gamma \in V$, $f(n, \bot, \top, \gamma)$ (the value returned by the alpha-beta-gamma algorithm starting at node n with this gamma update function, $\bot$ as the initial value of α, $\top$ as the initial value of β) is $\mathsf{val}(n)$.

Motivation of using CI vectors

Intuitively, the CI vector at n contains information about worlds in which MAX cannot win against the best defence even under complete information; hence, it is a waste of

[14] "CI" stands for "complete information".

[15] For a non-Boolean vector game, the i-th component of the vector $\mathrm{CI}(n)$ is the best reward MAX can guarantee if they had complete information.

time to compute the components of a local strategy at a node in the subtree rooted at n that correspond to these losing worlds. Using CI vectors as gamma then has two effects at a node n of MIN:

- $\mathrm{CI}(n)$ is used to mask the initialisation of the value at n, which is then used to update the value of β; therefore, β becomes a sharper upper bound and potentially provokes an earlier alpha-beta cut, thereby reducing the effective branching factor;
- $\mathrm{CI}(n)$ projects the computation of $\mathsf{val}(n)$ to the restricted universe $U \cap \mathrm{CI}(n)$. This makes the effective size of the universe smaller *at this node*[16], especially after an error from MAX or a killing move from MIN that makes MAX's winning support under complete information diminish largely.

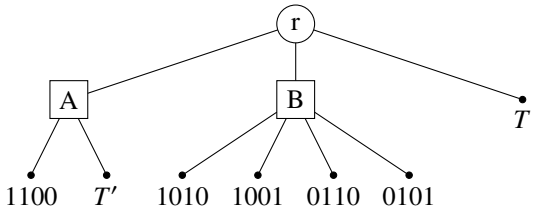


Figure 6.7: A vector game with 4 worlds, where T and T' denote two unspecified subtrees. Rewards vectors are written in a compact inline fashion.

Example. *To illustrate the effects of using CI vectors as gamma, consider the vector game in Figure 6.7. Suppose that* $\mathrm{CI}(r) = 1100$ *(which can achieved by altering the rewards in the subtrees T and T′). In addition, suppose that both the initial* β *and the initialisation I for the root are* $\top$*. Then before visiting the first child of r,* v[17] *is given* $\{\mathrm{CI}(r)\} = \{1100\}$ *as value, and* β *is also updated to be* $\{1100\}$.

- *At node A, a cut happens after visiting the first child; the whole tree T′ is pruned.*
- *At node B, the value returned for child B is* $\mathsf{val}(B) = \{1010, 1001, 0110, 0101\}$ *since there is no cut. Then at r, v is updated to be* $\{1100\} \sqcap \mathsf{val}(B) = \{1000, 0100\}$. *Notice how the differences between the vectors in* $\mathsf{val}(B)$ *in the components corresponding to losing worlds in* $\mathrm{CI}(r)$ *are erased by the meet with* $\mathrm{CI}(r)$*: it is as if we projected each returned value at r to the first two worlds. As a result, the size of families involved is reduced, since incomparable vectors in U can be comparable in the effective universe* $\mathrm{CI}(r) \subseteq U$*. In this particular example, the size of v after visiting child B is reduced from* 4 *to* 2*, which will also accelerate the computation between the meet of v and the returned value from the subtree T.*

Since the size of the universe and the branching factor are two main ingredients of the combinatorial explosion in Ginsberg's algorithm, CI vectors are important sources of optimisation. What makes CI vectors even more interesting to consider is that the computation of CI vectors for a CGII is linear-time, which means the time spent on CI vectors will usually be dwarfed by the running time of Ginsberg's algorithm itself.

Finally, it is also worth mentioning that the CI vectors at the children of a node can be used as a heuristic for move ordering. For instance, at a MIN's node, we may visit

[16] But not at all nodes in the subtree rooted at n; see discussions later in Subsection 6.5.4.

[17] v is the cumulative value at r for computing $\mathsf{val}(r)$ in Algorithm 6.

children with a CI vector of smaller cardinality first. The intuition is that these children correspond to MIN's best actions at this node when MAX has complete information; hence they may very well also be MIN's best actions when MAX only has incomplete information.

6.5.3 Information-revealing CGIIs

Deduction about the real world

We have already discussed in Subsection 5.4.2 the fact that in general, in games with incomplete information and public actions, MIN can have different available actions depending on the real world ω (or equivalently on their own type). For example, in card/tile-based games such as Bridge or Mahjong, MIN cannot play a card/tile that is not in their hand.

In such a game, MAX can deduce information about MIN's type by observing the action taken by MIN. For instance, in the game with incomplete information in Figure 5.7 (page 92), if MAX observes that MIN takes action l, then MAX knows that MIN cannot be of type C since a MIN of this type has no action l.

Back in Subsection 5.4.2, we have shown how to transform EFGs with one-sided incomplete information and public actions such as the one in Figure 5.7 into a vector game (cf. Figure 5.8) by vectorisation. The vector game obtained has reward vectors with components in $\{0, 1, *\}$. Hence, by taking $*$ to be 1, we get a Boolean vector game with the same maxmin value as the initial EFG. However, $*$ contains information about worlds that are impossible with the observations leading to a node; we can exploit such information to get more tree prunings.

We can formulate a notion of information-revealing CGIIs to encompass CGIIs in which such kind of deduction about MIN's type is possible for MAX. However, it does not harm to generalise the notion a bit to include even more different CGIIs, which is what we plan to achieve in the following paragraphs.

Incompatible belief state and revelation of information

Definition 6.5.5 (Incompatible belief state)**.** *Let G be a CGII. Let $*$ denote the largest reward MAX can receive at any leaf and in any world.*[18] *The* incompatible belief state *(abbreviated as IBS in the following) for MAX at a node $n \in \mathrm{N}(T)$ is defined to be*

$$\mathrm{IBS}(n) = \{\omega \in U \mid \textit{MAX receives } * \textit{ in } \omega \textit{ at all leaves reachable from } n\} \subseteq U.$$

Example. *If we assume the total order $0 < 1 < *$, then in the vector game in Figure 5.8 with universe $U = \{A, B, C\}$, we have $\mathrm{IBS}(\{A, B, C\}) = \emptyset$, $\mathrm{IBS}(\{A, B\}) = \{C\}$ and $\mathrm{IBS}(\{B, C\}) = \{A\}$.*

Remark. *Notice that in the previous example, the IBS at a node is the complement of the name we give to the node itself. This is not a coincidence, since we have named the nodes in Figure 5.8 with the* belief state *of MAX at each node.*

If we define the belief state of MAX at a node $\mathrm{BS}(n)$ in the traditional sense (i.e. the set of all worlds that are compatible with all the observations in the history), then in a vector game obtained by vectorisation, $\mathrm{IBS}(n)$ and $\mathrm{BS}(n)$ always form a partition of U, whence the name "incompatible belief state".

[18] In particular, in a vector game, $*$ is the largest component of all reward vectors.

Although IBS is defined to capture the notion of incompatible worlds with observations, it is actually more general than that: it contains information about MIN's dominated actions. For example, in the vector game in Figure 6.8, we have $* = 1$. Notice that $\text{IBS}(A) = \{\omega_1\}$, since MAX always receives 1 as reward in ω_1 if node A is ever reached, no matter how MIN plays in the following. This shows that picking a in ω_1 is a strictly dominated strategy for MIN. Hence, for the sake of computing the maxmin value of this game, we may assume that MIN of type ω_1 never takes this action; it is as if MIN has no action a in ω_1. Similarly, $\text{IBS}(B) = \{\omega_3\}$ since reaching B (by picking b) in ω_3 leads to a guaranteed maximum reward for MAX.

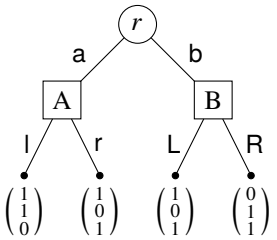


Figure 6.8: A vector game with a universe of 3 worlds.

In summary, the IBS at a node not only contains worlds that are incompatible with past observations leading to this node, but also worlds in which MIN has no strict incentive to take actions leading to this node.

Notice that the notion of IBS is well-defined for an arbitrary CGII and for both MAX and MIN. In the following, however, we only focus on IBS for MAX in CGIIs under the best-defence model, i.e. vector games. We will use the term *information-revealing* CGIIs when we have vector games obtained by vectorisation (hence with 0, 1, and $*$, as possible rewards for MAX) in mind, since in such games actions of MIN do reveal information about their hidden type. However, one should keep in mind that optimisations for information-revealing CGIIs involving IBS are actually applicable to all vector games due to the well-definedness of IBS for all vector games.

Computation of IBS

By definition of IBS, at an internal node n, for all children n' of n, we have $\text{IBS}(n) \subseteq \text{IBS}(n')$. This is rather intuitive when we think about information-revealing game, since MAX cannot have less information when they observe more.

In principle, for all nodes n in a vector game, $\text{IBS}(n)$ can be computed in $O(t|T|)$ time, where t is the size of the universe, by finding $*$ first (which takes $O(t|T|)$ time) then verifying if in a world all leaves reachable from n yields $*$ for MAX (which takes $O(|T|)$ time for each world).

However, in practice, $\text{IBS}(n)$ can be computed in a much more efficient way using game rules. For example, observing MIN play a card rules out all worlds in which they do not have this card. And $\text{IBS}(n)$ can usually be computed incrementally: given $\text{IBS}(n)$ for a node n of MIN, for a child n' of n reached by an action a of MIN, we may compute $\text{IBS}(n')$ in $O(t)$ time[19] using $\text{IBS}(n')$ and a, thus saving a factor of $O(|T|)$, which is usually enormous.

[19] Or even better than linear time, if subsets of the universe are represented in a compact data structure that is relatively easy to update, such as Boolean binary diagrams.

Combining CI and IBS in the alpha-beta-gamma algorithm

Just as CI vectors provide admissible initialisation for MIN's nodes, IBS vectors provide admissible initialisation for MAX's nodes. Indeed, at a MAX's node n, the vector that has $*$ as components for worlds in $\mathrm{IBS}(n)$ and 0 as components for all other worlds is a lower bound for the support of all MAX's local strategies at n, for the simple reason that a world in $\mathrm{IBS}(n)$ guarantees a reward of $*$ for MAX at all leaves reachable from n.

Therefore, by using CI and IBS together, we may have more or earlier alpha-beta cuts in Algorithm 6; we will examine the nature of these cuts in the following paragraphs.

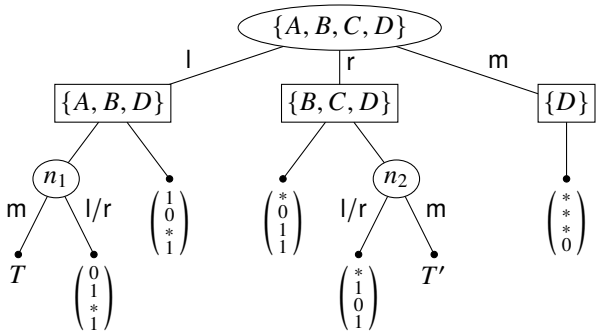


Figure 6.9: An information-revealing CGII, where T denotes an unspecified subtree in which MIN of type D can force a loss for MAX and T' denotes another unspecified subtree in which MAX can force a win.

Consider the vector game in Figure 6.9. There are 4 types of MIN, labelled by $\{A, B, C, D\}$. In this game, only MIN of type D has action m; MIN of type A has no action r; MIN of type C has no action l.

To facilitate the visualisation of this game, we name the nodes in the first two layers by the belief state of MAX at these nodes. For instance, if the node labelled by $\{D\}$ is reached, then the belief state of MAX is $\{D\}$ since only MIN of type D has action m; equivalently, $\mathrm{IBS}(\{D\}) = \{A, B, C\}$.

Let us assume $1 < *$ for the computation of CI vectors. Then the right child of n_1 (i.e. the root of T, denoted by $r(T)$) has $(*, *, *, 0)$ as its CI vector; $\mathrm{CI}(n_1) = (0, 1, *, 0)$, which means that starting from node n_1, if MIN is of type B, MAX under complete information can force a win; if MIN is of type A or D, MIN can force a loss for MAX starting from n_1. If MIN is of type C, then n_1 is actually not reachable since MIN of type C has no action l, hence the $*$ in $\mathrm{CI}(n_1)$. In set representation, the CI vector at n_1 reads $\{B, C\}$. On the other hand, $\mathrm{IBS}(n_1) = \{C\}$, and $\mathrm{IBS}(r(T)) = \{A, B, C\}$.

Now, suppose that Algorithm 6 traverses the nodes of the same layer from left to right, hence visits the left child first at n_1. Then at $r(T)$, the value of beta is inherited from the value of beta at n_1 after being updated by $\mathrm{CI}(n_1)$, hence is at most $\{B, C\}$; the value of α is the value of α from the parent of n_1 updated by $\mathrm{IBS}(r(T))$, hence is at least $\{A, B, C\}$. As a result, an alpha-beta cut happens and the whole tree of T is pruned.

CI action pruning

What is the intuition behind the cut above? Well, $\mathrm{IBS}(r(T))$ tells us that MIN of types A, B, and C have no incentive to pick actions leading to the node $r(T)$; $\mathrm{CI}(n_1)$ tells us

that the only type not in $\mathrm{IBS}(r(T))$, i.e. type D, can force a loss for MAX. Intuitively, the cut at $r(T)$ is equivalent to saying that at n_1, it is unnecessary to consider the possibility that MIN takes action m, since all types of MIN that have incentive to take this action (D in our case) can force a loss for MAX; this latter fact has already been taken into account by initialising the value of n_1 with $\mathrm{CI}(n_1)$, hence the tree following action m can be safely pruned without affecting the computation of the maxmin value.

More generally, at a node n of MIN in an information-revealing CGII, one can safely skip all MIN's actions that are only available (dictated by the IBS vector at a child of n, which contains all types of MIN that cannot reach this child since the action leading to this child is not available to MIN of those types) in worlds in which MAX cannot win even under complete information (dictated by $\mathrm{CI}(n)$). We call this particular type of game tree pruning, which is automated by using CI and ISB together in Algorithm 6, *CI action pruning*. The soundness of this pruning is guaranteed by the correctness of Algorithm 6 (cf. Proposition 6.5.2) and the fact that CI vectors are admissible gammas while IBS vectors form an admissible heuristic function.

This pruning is particularly useful for real-life games with incomplete information in which MIN's available actions largely depend on their type (e.g. card games such as Bridge); for these games, it can effectively reduce the branching factor at MIN's nodes.

6.5.4 Prunings with path winning supports

Motivation

Notice that CI action pruning is especially efficient at nodes where MAX only has a slime chance of winning (i.e. when the size of the winning support under complete information is small compared to the size of the universe), which is typically the case after MAX commits some errors. Hence, generally speaking, CI action pruning makes the search algorithm spend less time in computing the situational values in subtrees following MAX's bad moves, and focus on other (more promising) subtrees, thereby improving its performance. What about MIN's bad moves? Can we also accelerate the tree search after some errors from MIN? This is the subject of the current subsection.

Keen readers may have noticed that in the CGII in Figure 6.9, at the root MIN of type D can force a loss for MAX by playing m to end the game immediately, which means $D \notin \mathrm{CI}(r)$, where r is the root, which implies that D is not in any vector from $\mathsf{val}(r)$.[20]

Therefore, we may wish to assume that if MIN has not played m at the root, then they are not of type D; after all, we know that the exact value for the component corresponding to type D in supports for MAX's local strategies is inessential for computing the maxmin value. This assumption allows the algorithm to search with a smaller effective universe at all subtrees and to prune more branches. For example, the subtree T' following branch m from node n_2 should be pruned since MIN could not be of type D at n_2 and MIN of the other types has no action m.

However, this pruning is not encompassed by CI action pruning. For instance, $D \in \mathrm{CI}(n_2)$, hence the action m at n_2 will not be pruned by the algorithm. The main reason behind this is that CI vectors are purely *local* objects (i.e. the CI of a node

[20] Recall from Lemma 6.5.4 that the CI vector at a node dominates all vectors in the situational values of the same node (i.e. $\mathsf{val}(n) \preceq \{\mathrm{CI}(n)\}$). Another way to understand this result in our example is the following: at the root, the algorithm computes (during the computation of $\sqcap$) the bit-wise minimum between the support $(*, *, *, 0)$ from the branch m and supports for MAX's local strategies backed up from other subtrees, thereby causing the component corresponding to type D in all resulting supports to be 0.

depends solely on the subtree rooted at this node): $\mathrm{CI}(n_2)$ contains no information about the fact that D is actually not in $\mathrm{CI}(r)$.

From another perspective, after MIN's bad moves (e.g. MIN of type D plays r at the root, which is an error since there is now a strategy for MAX to force a win against MIN of type D), the CI vector actually contains more worlds since MAX can now win under complete information against more types (in the example above, type D); the search algorithm then spends more time than necessary by adding back some types to the effective universe at a node n, although MIN of these types can force a loss for MAX at nodes that are ancestors of n.

PWS and its properties

To fix this undesirable behaviour of CI action pruning, we introduce a *non-local* version of CI that can remember information from the past about types of MIN that can force a loss for MAX.

Definition 6.5.6 (Path winning support)**.** *The* path winning support *(for MAX under complete information) is defined recursively in a top-down manner by* $\mathrm{PWS}(r) = \mathrm{CI}(r)$ *and* $\mathrm{PWS}(n) = \mathrm{PWS}(n') \cap \mathrm{CI}(n)$*, where* n' *is the parent of* n*.*

In the following, we refer to $\mathrm{PWS}(n)$, the path winning support at a node n, as the *PWS vector* at n.

Example. *In the CGII in Figure 6.9, we have* $\mathrm{CI}(r) = \{A, B, C\}$*,* $\mathrm{CI}(\{B, C, D\}) = \{A, B, C, D\}$*, and* $\mathrm{CI}(n_2) = \{A, B, D\}$*. Hence,*

$$\begin{aligned} \mathrm{PWS}(r) &= \mathrm{CI}(r) = \{A, B, C\}, \\ \mathrm{PWS}(\{B, C, D\}) &= \mathrm{PWS}(r) \cap \mathrm{CI}(\{B, C, D\}) = \{A, B, C\}, \\ \mathrm{PWS}(n_2) &= \mathrm{PWS}(\{B, C, D\}) \cap \mathrm{CI}(n_2) = \{A, B\}. \end{aligned}$$

Notice that $D \notin \mathrm{PWS}(n_2)$ *as* D *is not in the PWS vector at the root, an ancestor of* n_2*.*

It is clear that the PWS is always a subset of CI.

Lemma 6.5.7. *For all* n*,* $\mathrm{PWS}(n) \subseteq \mathrm{CI}(n)$*.*

In addition, PWS is decreasingly monotone in terms of inclusion along all paths of the game tree starting from the root.

Lemma 6.5.8. *Let* n *be an internal node and* n' *be a child of* n*. Then* $\mathrm{PWS}(n') \subseteq \mathrm{PWS}(n)$*. Moreover,* $\mathrm{PWS}(n) = \mathrm{PWS}(n')$ *if* n *is a node of MIN.*

Proof. $\mathrm{PWS}(n') \subseteq \mathrm{PWS}(n)$ follows directly from the definition. If n is a node of MIN, then $\mathrm{CI}(n) \subseteq \mathrm{CI}(n')$ since $\mathrm{CI}(n)$ is defined as the intersection of the CI vectors at all the children of n. Hence, $\mathrm{PWS}(n) \subseteq \mathrm{CI}(n')$ and $\mathrm{PWS}(n') = \mathrm{PWS}(n) \cap \mathrm{CI}(n') = \mathrm{PWS}(n)$. □

Remark. *The fact that a node of MIN and all its children have the same PWS vector shows again the non-locality of PWS: each child of a node of MIN has access to some information from its siblings via this common PWS vector.*

Since $\mathsf{val}(n) \preceq \{\mathrm{CI}(n)\}$ for all nodes n, we know that $g(n, \gamma) = \gamma \sqcap \{\mathrm{CI}(n)\}$ is a path-admissible gamma update function. In particular, $g(n, \gamma) = \{\mathrm{PWS}(n)\}$ is path-admissible, which is equivalent to using the previous gamma update function in

Algorithm 6 with $\gamma = \top$ as the initial gamma. Therefore, PWS vectors can be used as gamma in Algorithm 6. Concretely, by Corollary 6.5.3, $f(r, \bot, \top, \top)$ (the value returned by the alpha-beta-gamma algorithm with this gamma update function, $\bot$ as the initial value of alpha, $\top$ as the initial value of beta and gamma) is $\mathsf{val}(r)$.

PWS prunings

Just as CI vectors, PWS vectors are computable in linear time and can also be used as heuristics for move ordering. We now have a look at how PWS vectors produce more alpha-beta cuts.

Let us write $g_{\mathrm{CI}}(n, \gamma) \coloneqq \mathrm{CI}(n)$ and $g_{\mathrm{PWS}}(n, \gamma) \coloneqq \gamma \sqcap \{\mathrm{CI}(n)\}$ for the gamma update function corresponding to using CI vectors and PWS vectors, respectively, as gamma in Algorithm 6.

Recall that using g_{CI} in Algorithm 6 has three effects at a node n of MIN:

1. $\mathrm{CI}(n)$ is used to make β sharper to trigger an alpha-beta cut earlier;
2. $\mathrm{CI}(n)$ is used to reduce the effective size of the universe at n;
3. $\mathrm{CI}(n)$, with α at the children of n updated by their IBS vectors, creates CI action pruning, i.e. pruning of branches corresponding to actions that are only available to types against which MAX cannot win.

We refer to all these effects brought by using g_{CI} (and IBS vectors as heuristic function for initialisation of MAX's nodes) as *CI prunings*. We will examine one by one how using g_{PWS} instead of g_{CI} affects these prunings.

Effect 1 Since $\mathrm{PWS}(n) \subseteq \mathrm{CI}(n)$ (cf. Lemma 6.5.7), it is clear that using g_{PWS} produces no fewer nor later alpha-beta cuts than using g_{CI}.

Effect 2 Using g_{PWS} reduces the effective size of the universe *not only* at a node n of MIN, but at all nodes (including those of MAX) in the subtree rooted at n. This is a direct consequence of the monotonicity of PWS vectors (cf. Lemma 6.5.8). Indeed, all the initial values of the nodes in the subtree rooted at n are masked by a gamma that is no larger (in terms of $\preceq$) than $\{\mathrm{PWS}(n)\}$. In particular, this is also true at the leaves in this subtree; hence the value returned for these leaves is also masked by a gamma no larger than $\{\mathrm{PWS}(n)\}$.

As a result, the search algorithm has $\mathrm{PWS}(n) \subseteq U$ as its effective universe in this subtree instead of the whole U, which means PWS is even more efficient than CI in reducing search time in branches after MAX's errors or MIN's good moves.

Effect 3 *PWS action pruning* is also more efficient than CI action pruning, since $\mathrm{PWS}(n) \subseteq \mathrm{CI}(n)$ for all nodes n.

For the last point, take a look again at the example in Figure 6.9. CI action pruning correctly prunes T by ignoring action m at n_1, but it does not prune the subtree T'. On the other hand, PWS action pruning correctly prunes both T and T'. To verify that T' is indeed pruned, notice that $\mathrm{PWS}(n_2) = \{A, B\}$ while the IBS vector at the root of T' reads $\mathrm{IBS}(r(T')) = \{A, B, C\}$; an alpha-beta cut is then triggered at the beginning of the search at $r(T')$.

We refer to all these prunings as *PWS prunings*. We have observed that PWS prunings are a net improvement over CI prunings: not only do PWS prunings accelerate

the game tree search after MAX's bad moves or MIN's good moves by considering a smaller effective universe, they also do not have the illusion that MAX can win against certain types of MIN at the root after seeing bad moves from these types of MIN (e.g. type D in previous examples).

In short, PWS offers better guidance for game tree search by taking advantage of a pre-analysis (the computation of CI and PWS vectors at each node) that takes practically negligible time compared to the tree search itself (linear time vs. NP-hard) in order to, when searching in some part of the game tree, incorporate non-local information about other parts of the tree, thereby combatting the effects of non-locality (notably the combinatorial explosion).

As we have mentioned before at the beginning of Subsection 6.5.2, CI and PWS prunings are inspired by human Bridge players. For example, if a player finds out that they cannot possibly win if certain hidden cards are not in a certain hand (of their opponent or partner), then they will assume that these hidden cards are indeed in the right hand for their win to be possible, and continue their offline planning based on this assumption.[21] This amounts to using $\mathrm{CI}(r)$ as the effective universe at every node $n \in \mathrm{N}(T)$, which is the case when g_{PWS}[22] is used in the alpha-beta-gamma algorithm. And ideas similar to PWS vectors (i.e. not adding losing worlds at an ancestor node back to the effective universe after MIN's bad moves) are also used to avoid pitfalls caused by non-locality. For a Bridge deal with non-locality that can be avoided by using PWS prunings, see Deal 1 (page 168) in Appendix A.4.1.

Caveat of using PWS: cache problem

However, there is a price to pay for the enhanced performance brought by PWS prunings: they are not naturally compatible with caching as CI prunings do. Concretely, if the game we consider has a DAG instead of a tree, then the PWS vector at a node depends on which path we take to arrive at this node.

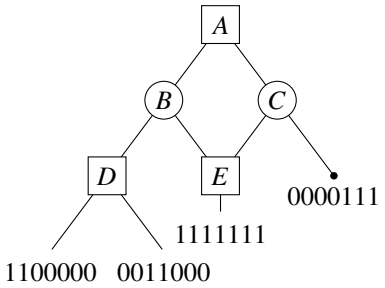


Figure 6.10: A vector game as a DAG with 7 worlds, where rewards vectors are written in a compact inline fashion.

Consider the game in Figure 6.10. In this game, the CI vectors are $\mathrm{CI}(A) = \mathrm{CI}(E) = 1111111$, $\mathrm{CI}(C) = 0000111$, and $\mathrm{CI}(B) = \mathrm{CI}(D) = 1111000$. Let us examine the execution of the alpha-beta-gamma algorithm with g_{PWS}, $\alpha = \bot$, $\beta = \gamma = \top$, and trivial initialisation.

[21] This search technique is called *hypothèse de nécéssité* (hypothesis of necessity) in the French literature on Bridge.

[22] Note that we are really referencing g_{PWS} here instead of g_{CI}: if g_{CI} is used, then the effective universe is reduced to $\mathrm{CI}(r)$ only at r, *not* at every node.

Suppose that the game tree search algorithm explores the child B of node A first.[23] At vertex E, the PWS vector reads $\mathrm{PWS}(E) = \mathrm{CI}(A) \cap \mathrm{CI}(B) \cap \mathrm{CI}(E) = 1111000$. Hence, the game tree search algorithm returns $\{1111111\} \sqcap \{\mathrm{PWS}(E)\} = \{1111000\}$ as the value of E, which is then used to compute the value returned for B, which is $\{1100000, 0011000\}$.

If the result $\{1111000\}$ is directly cached for node E, then when the game tree search algorithm descends again at E via the path $A - C - E$, it will take 1111000 as the value of E. As a result, the value of C is computed to be $\{1111000\} \sqcap \{0000111\} = \{0000000\}$. Therefore, the algorithm erroneously concludes that the value of the root is $\{1100000, 0011000, 0000000\}$ and the maxmin value of the game is 2. Indeed, this result is not correct as the third vector in the situational value of the root (i.e. the one corresponding to taking the action leading to C at the root) is actually 0000111 and the maxmin value is in fact 3.

The origin of this problem is, as we have stated before, that the PWS vector at a node in a DAG depends on which path we take. In the example above, there are two paths leading to E, the first one $A - B - E$ with PWS vector 1111000 and the second one $A - C - E$ with PWS vector 0000111. One can see that these two PWS vectors are incomparable (with set inclusion as the partial order). The first PWS only deems the first 4 worlds to be useful for computing the situational value of the ancestors of E (along the path $A - C - E$). Hence, the value of E computed when following the path $A - B - E$, which is the projection of $\mathsf{val}(E)$ over the first 4 worlds, offers no useful information for the path $A - C - E$, for which the PWS vector deems only the last 3 worlds to be useful.

Notice that CI vectors do not suffer from this problem, since they are local and depend solely on the subtree rooted at a node; they are naturally compatible with caching schemes.

A naïve attempt to fix this cache problem by recording the PWS vectors we use to compute the value at a node in the cache does not work, since there are examples of vector games with a DAG which have an exponential (in the number of MIN's types) number of incomparable PWS vectors at some nodes. At the moment of writing, it is still an open question how to design a caching scheme for the alpha-beta-gamma algorithm that is compatible with PWS vectors without the need to store every PWS vector at a node.

6.6 Conclusion

In this chapter, we have presented different exact optimisations for Ginsberg's algorithm from two categories: strategy pruning and game tree pruning. Strategy pruning reduces the size of families involved in the computation of situational values; a notable example is elimination of dominated strategies. Game tree pruning reduces the effective branching factor, and in some cases also reduces the effective size of the universe (e.g. PWS prunings).

Prospectives

There are many possible ways to pursue the study in this chapter. For instance:

[23] Notice that if move ordering is carried out using CI vectors as heuristics, then at A child B will be chosen first since $|\mathrm{CI}(B)| > |\mathrm{CI}(C)|$.

- design a caching scheme that can make full use of PWS prunings;
- conduct experiments on a better constructed benchmark to compare the effects of different exact optimisations;
- investigate non-exact optimisations for Ginsberg's algorithm (e.g. techniques such as Monte Carlo tree search);
- consider using appropriate data structures (e.g. Epistemic Splitting Diagrams from Niveau and Zanuttini (2016)) to accelerate the computation of the operators from the family algebra (union, meet, maximal elements, etc.).

Chapter 7

Opponent models and recursive reasoning

7.1 Introduction

In this chapter, we first take a look again at the best-defence model, under which we have developed many algorithmic ideas (cf. Chapter 6). In some cases, the best-defence model is well justified, either because MIN has complete information, or MIN does not have too much incomplete information (i.e. the equivalence classes of MIN do not contain too many worlds).

For example, in the game of Bridge, MIN typically has far less incomplete information than MAX, since MAX often reveals information about their hidden hand during the bidding phase to become declarer. Hence, human Bridge players typically find their first robust plan for gameplay in an offline fashion (i.e. before playing their first card) under the best-defence model. Actually, the fact that this model is widely used by Bridge experts is the initial motivation for Frank and Basin (1998) to formalise this model.

However, in Section 7.2, we show some examples in which the best-defence model is not appropriate. We then have an informal brainstorming to discuss ideas that allow us to go beyond the best-defence model while squeezing out the last drop of what it can offer to use.

In Section 7.3, we thoroughly explore one of the ideas discussed in Section 7.2: opponent-model search applied to CGIIs under the best-defence model. Such search allows taking into account the possibility that the behaviour of MIN is not completely nondeterministic, and that MIN's reasoning can be simulated by MAX.

Then in Section 7.4, which concerns the idea of recursive opponent modelling, we give a general framework of such modelling, which encompasses many instances of recursive modelling in the literature. We then show how such modelling can be applied to CGIIs under the best-defence model using the OM search algorithms presented in Section 7.3. Such a combination (recursive modelling with OM search under the best-defence model) captures nicely how expert Bridge players go beyond the best-defence model in their planning for card play; we show an example to illustrate this point to conclude this section.

The content in this chapter is still a work in progress. As a consequence, this chapter is less well-shaped than the previous chapters. We are still ruminating on how

to formalise the ideas and formalisms we touch upon in this chapter and to apply them to a broader field; ironing them out is one of the priorities of our future work.

7.2 Beyond the best-defence model

We have studied thoroughly algorithms for CGIIs under the best-defence model (Chapter 5) and optimisations for these algorithms (Chapter 6). However, contrary to what the best-defence model assumes, MIN does not have complete information in general. In such situations, the best-defence model gives too much power to MIN, by adding pure strategies that they can implement only when they have complete information.

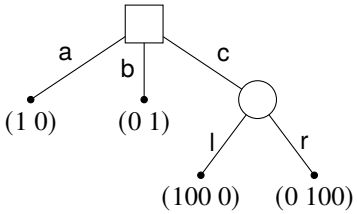


Figure 7.1: A CGII with two types of MAX and one type of MIN.

Example. *For example, consider the CGII in Figure 7.1, in which there are two types of MAX but one type of MIN; in other words, MAX has complete information while MIN only has single-agent incomplete information. Hence, the rewards are written horizontally, in contrast with rewards in vector games. With the uniform prior over MAX's types, one can verify that the (pure or behaviour) maxmin value for MAX of the game is* 50*, which is achieved by playing action* c*.*

However, the maxmin value for each type of player MAX under the best-defence model is 1*, which is obtained by playing* a *for the first type and playing* b *for the second type. The reason behind this is that under the best-defence model, MAX of the first type assumes that MIN knows their type and will play* r *to give MAX a payoff of* 0 *if MAX plays* c*; MAX of the first type therefore prefers (in the maxmin/security/robustness sense) playing* a*, which guarantees a payoff of* 1*. And a similar reasoning leads MAX of the second type to prefer* b*.*

The discrepancy between the two values (1 and 50) is due to the fact that MIN actually cannot pick between l and r according to MAX's actual type. This clearly motivates us to go beyond the best-defence model; the question now is "how".

Rigorously speaking, the best-defence model has two assumptions:

1. It assumes that MIN has complete information in a game.

2. It assumes that MIN may use any of their strategies that are legal under complete information.

In our previous work in Chapter 5 and Chapter 6, the first assumption is taken into account by considering only vector games, while the second one is taken into account by focusing on computing maxmin values against all pure strategies of MIN.

Behaviour maxmin

Let us first mention one obvious option that allows us to get rid of the best-defence model in some cases: consider behaviour maxmin instead of pure maxmin as solution concept.

Indeed, recall that PURE MAXMIN is NP-hard for CGIIs in which at least one team has incomplete information (Table 5.1, page 74), while BEHAVIOUR MAXMIN is in P for EFGs with perfect recall (Table 4.5, page 64), hence *a fortiori* for CGIIs in which both teams have incomplete information. In addition, allowing MAX to use behaviour strategies guarantees better payoffs, which seems to be the cherry on the cake.

However, this line of reasoning is a bit misleading; switching to behaviour maxmin is akin to burying one's head in the sand for the following reasons.

- When MAX has multi-agent incomplete information / multi-agent perfect recall, the complexity of PURE MAXMIN and the one of BEHAVIOUR MAXMIN is the same (compare Table 4.1 and Table 5.1 to Table 4.5). Indeed, if MAX is multi-agent while MIN is single-agent, then both decision problems are NP-hard; if both teams are multi-agent, then both problems are Σ_2^{P}-hard. Hence, switching to behaviour maxmin does not seem so promising when we tackle team games.
- Sometimes we are forced to adopt pure strategies for MAX. It can be because pure strategies are desirable for their deterministic nature or for MAX's lack of sources of randomness. But the strongest argument comes from the need to model human players, who have been experimentally shown to be inclined to reason in terms of pure strategies rather than mixed or behaviour strategies (Dhami, 2019, Chapter 1). If the MAX in consideration is a human player, and we wish to simulate their reasoning (in order to collaborate with them or defeat them), then again we have to resort to algorithms for finding optimal pure strategies.

Opponent model

To go beyond the best-defence model, we first turn to the concept of opponent models, a general discussion on which can be found in Section 2.5. For various reasons, MAX needs not assume that MIN may use any of their pure strategies as in the best-defence model (i.e. assumption 2), in particular if MAX has a model of MIN's behaviour or reasoning so that they can partially predict what MIN would do in a game.

We list below a few of these reasons for taking an opponent model into account, which are by no means comprehensive.

- MIN is a player with limited computational resources.
- MIN is not a rational player (which is often the case for human players).
- MIN is rational but has a different utility function than prescribed by the game (e.g. when the payoffs are measured in money, but MIN's subjective reward is an unknown and non-linear function of the amount of money they receive).
- MAX has past experience of playing against MIN, hence has accumulated knowledge about MIN's playing style (e.g. Federer vs. Nadal in their head-to-head matches).
- MAX is somehow aware of how MIN reasons (e.g. teachers are aware of usual mistakes that students would make in some subjects, since they partly understand the thinking process of their students when facing new material).

- When MIN is multi-agent, the coordination between agents of MIN may be common knowledge due to game rules.

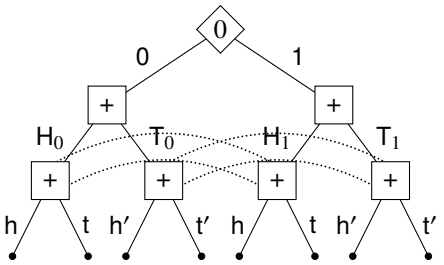


Figure 7.2: Matching Pennies with multi-agent incomplete information for MAX, with rewards omitted.

Example. *Let us illustrate the last point with the CGII (represented by its corresponding EFG of chance) in Figure 7.2. In this game, there are two agents (1 and 2) of MAX: MAX 1 has complete information (and can pick between heads and tails according to the bit chosen by Nature), while MAX 2 has incomplete information (and can pick between heads and tails according to the choice of MAX 1 but not to the bit chosen by Nature).*

Suppose that the rewards are such that the agents of MAX must pick the same face when Nature chooses 0*, and different faces when Nature chooses* 1*. Then one can verify that the pure (but also behaviour) maxmin strategies are the following: MAX 1 uses their action to tell MAX 2 what Nature's choice is, so that MAX 2 can pick the right face accordingly. For example, MAX 1 plays heads* H_0 *and tails* T_1 *in world* 0 *and* 1*, respectively, and MAX 2 plays* h *when MAX 1 plays heads (the same face as MAX 1 since they are in world* 0*) and* h' *when MAX 1 plays tails (different face from MAX 1 since they are in world* 1*).*

During the card play phase of Bridge, two agents of MIN have to transmit information about their own hand to each other so as to play the right strategy to defeat MAX. The situation is very similar to the example above: one agent of MIN encodes some information into the card that they choose to play so that another agent of MIN can choose the right strategy accordingly. Such a way to encode and transmit information is referred to as *signalling*. However, in the rules of competition of Bridge, all conventions must be common knowledge, including signalling conventions between every pair of players. Hence, when MAX sees MIN 1 play a certain card, in principle they would be able to deduce the same information as MIN 2. An example of from Bridge is given by Deal 2 (page 168) in Appendix A.4.1.

To automate this kind of reasoning and deduction based on knowledge about MIN's conventions, we again need to take opponent models into account.

Remark. *It is now clear that for some of these situations, the validity of not only the best-defence model but also the maxmin value (regardless of pure or behaviour) itself is questioned. For example, pure maxmin as a solution concept in a zero-sum EFG with perfect information is equivalent to assuming common belief in future rationality (cf. Perea (2012, Chapter 8)), which in particular implies that MIN is not just rational but in fact infinitely rational; this assumption does not hold in many cases evoked above.*

Recursive opponent modelling

We have already argued that opponent models are important resources to get rid of assumption 2 of the best-defence model. Now we have a brief look at one idea inspired again by human gameplay in Bridge about how to circumvent assumption 1.

Take a look again at the game in Figure 7.1. Pure maxmin under the best-defence model overestimates MIN's power at their decision node by assuming that they can pick between l and r according to MAX's type. To partially solve this problem, human Bridge players employ a recursive opponent modelling.

Informally, according to this model, each player at level-0 reasons under the best-defence model for their own type. Then, a player at level-1 assumes that their opponent plays their level-0 strategy. For example, in the game in Figure 7.1, MAX at level-1 computes the maxmin strategies under the best-defence model *for* MIN at level-0. By doing so, MAX realises that MIN is actually indifferent between the actions l and r. If MAX assumes the strategy played by MIN at level-0 will be uniformly random between these two actions,[1] which is a rather reasonable assumption (Camerer et al., 2004). Therefore, MAX at level-1 now prefers action c.

This type of iteration can go on forever, but usually human players only reason recursively at a depth of 2 or 3 (Dhami, 2019). Indeed, near the end of this dissertation, we will study a simplified version of a real Bridge deal involving a recursive reasoning at level-3 (Figure 7.4); such deals are extremely rare even in Bridge championships, which corroborates the fact that human players have a very limited depth for recursive reasoning.

Notice that such recursive reasoning is a form of reasoning about opponent's strategies: if MIN plays s_{-}, this means that at a certain level s_{-} is optimal with respect to MIN's private information, from which MAX can deduce something about MIN's private information. Of course, this type of reasoning has no guarantee of soundness or robustness; MAX can very well make a wrong assumption about MIN's level or tie-breaking between indifferent strategies, which may have catastrophic consequences.

The idea of recursive opponent modelling is by itself very attractive, especially because it captures human strategic choices, which makes it particularly useful for designing AI systems that collaborate with or compete against humans.

In addition, recursive opponent modelling is also interesting from a complexity-theoretic perspective. If the real world is ω, then when each player reasons at level-1 under the best-defence model, it is as if they played in a CGII with the universe restricted to worlds that are indistinguishable by them from ω. At level-2, MAX, for example, considers MIN's reasoning in every world ω' that is indistinguishable from ω by MAX. To this end, MAX has to imagine what MIN imagines (at level-1, hence under the best-defence model) that MAX will do in worlds ω'' that are indistinguishable from ω' by MIN. One can see where this is going: at level-k, a player has to consider all worlds reachable from ω within k alternations of the equivalence relations in the Aumann model of the CGII. Therefore, in CGII with large and complicated Aumann model but very local equivalence relations (i.e. only worlds that are really close are

[1] We emphasise that this is not conceptually equivalent to assuming that MIN plays a mixed strategy at level-0. In fact, the assumption is rather to assume that there are players in the position of MIN at level-0 who, for whatever reason, pick l; and there are also players who pick r. Since MIN at level-0 is indifferent between these two actions, there are intuitively the same number of people who pick l as of those who pick r. This perspective, coming from epistemic game theory (cf. Perea (2012)), provides a new interpretation of mixed strategies: it is not a fixed player who randomises between strategies, but a population of players having different preferences over the strategies; since we have no idea of the exact preference of a particular opponent, it is as if they would pick strategies randomly.

indistinguishable by players), then k such alternations only cover a small part of the whole universe.

In particular, we can see that the idea of recursive opponent modelling is a generalisation of the best-defence model. We leave the formal definition of recursive opponent modelling, its properties in terms of the Aumann model in consideration, and its logical foundation to future work.

7.3 Opponent-model search in vector games

The content in this and the following section is to be published in AAAI 2024 (Li et al., 2024).

7.3.1 Introduction

Opponent models are models that describe or predict how an opponent reasons or behaves in a game. Such models have been explicitly incorporated into game tree search algorithms (e.g. minimax, $\alpha\beta$ search, MCTS) for games with perfect information (Iida et al., 1993, 1994; Sturtevant and Bowling, 2006; Sturtevant et al., 2006) to find robust strategies against a given opponent model, i.e. strategies that guarantee a given payoff against any strategy deemed possible by the opponent model. The knowledge of opponent models can accelerate the game tree search (e.g. by pruning branches not considered by an opponent) and improve the performance of the strategies obtained (e.g. by exploiting the weakness of an opponent).

In this section, we extend the idea of opponent-model search to (two-player, zero-sum) games with incomplete information. We first propose different ways of taking opponent models into (which is of interest beyond the setting of incomplete information), and give algorithms for computing the corresponding robust strategies for such games. We then propose a principled method for taking into account a probability that the opponent does not behave according to any of the given models. Finally, we show an application of these models to the recursive modelling of opponents, where a level-k player assumes that their opponent reasons at some level lower than k, and recursively down to level 0.

7.3.2 Problem setting

Throughout this section, we focus on zero-sum vector games, i.e. two-player zero-sum CGIIs with one-sided incomplete information;[2] such a CGII will be denoted by $G = \langle T, \mathrm{P}, t, \vec{u}, \vec{\rho} \rangle$ (Definition 5.4.1, page 91).

Recap on maxmin value without OM

We first quickly recap the computation of maxmin value without OM so that we can contrast it with the computation of maxmin value with OM.

[2] Our study can be easily extended to more players and general-sum, with the exception of the lexicographic setting, for which the definition of opponent models does not trivially generalise.

Pure maxmin We already know that the pure maxmin value of a vector game is NP-hard to compute (cf. Proposition 5.3.3, page 76); Ginsberg's algorithm (e.g. without reduction: $\texttt{MiniMax}(\mathcal{P}_{<\infty}(\mathbb{R}^t), \mathsf{eval}, \cup, \sqcap)$) allows us to compute this value in the sense of Proposition 5.4.2 (page 97).

Example. *In the vector game in Figure 5.6 (page 92), the pure maxmin value is* $\frac{2}{5}$ *and is achieved by two pure strategies:* (l, L) *and* (r, R).

Mixed maxmin Contrary to pure maxmin, the mixed maxmin value can be computed in polynomial time with the linear-programming algorithm proposed by Koller and Megiddo (1992, Chapter 3).

In short, this algorithm relies on two insights. First, the set of all mixed strategies of MAX can be represented in sequence form, i.e. by a system L of linear equalities, with linearly many (in the size of the game tree) variables and equalities. Second, for every threshold v and every mixed strategy σ_+ of MAX represented as a solution to L, it can be verified in linear time whether $\min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}(\sigma_+, s_-) \geq v$ holds by computing MIN's best responses to σ_+. This computation serves as a separation oracle, under which a linear program (LP) maximising v with respect to the constraints in L computes the mixed maxmin.

Example. *In the vector game in Figure 5.6 (page 92), the optimal mixed strategy for MAX is the uniform strategy, i.e. a uniform distribution over all* 4 *pure strategies of MAX. This strategy yields an expected payoff of at least* $\frac{1}{2}$*, which is the mixed maxmin value and is better than the pure maxmin value* $\frac{2}{5}$.

The above algorithm has been improved by von Stengel (1996); Koller et al. (1996). However, for simplicity, we only show modifications of the initial algorithm for taking opponent models into account. Adapting them to the improved algorithms is straightforward.

Maxmin value against OMs

The setting of maxmin value in the presence of opponent models has already been studied from a complexity-theoretic perspective for EFGs in Section 4.4. The focus of the current section is on the algorithmic aspects: we aim to formulate algorithms for computing the maxmin value against a given set of opponent's strategies:

$$\underline{v}_+ \coloneqq \max_{\varsigma_+ \in \Sigma_+} \min_{\omega_- \in \Sigma^{\mathrm{O}}_-} \mathcal{U}(\varsigma_+, \omega_-),$$

where Σ_+ is the set of all pure or all mixed strategies of MAX, depending on the context; $\Sigma^{\mathrm{O}}_- \subseteq \Sigma^{\mathrm{B}}_-$ is a set of given behaviour strategies of MIN (also called *opponent models* or *OMs*). We will study OM search algorithms for both pure and mixed maxmin, albeit with a focus on the pure one, since algorithms for mixed maxmin only require minor modifications in the presence of opponent models.

In general, OMs are models of the opponent's reasoning, which can come in various forms (cf. references in Section 2.5). As a quite general setting, we consider that each OM describes a behaviour strategy of MIN, which are as expressive as mixed strategies in games with perfect recall, and *a fortiori* in CGIIs. For a strategy represented by a mixed strategy or another linear representation (e.g. sequence form (Koller and Megiddo, 1992), or by an evaluation function (Iida et al., 1993)), its equivalent behaviour strategy can also be computed in linear time.

Algorithmically, we assume that the OMs are given in the input and that each computation of $\omega_-(n, i, n')$ takes constant time, where $\omega_-(n, i, n')$ is the probability that under the OM strategy ω_-, MIN chooses $n' \in \mathrm{C}(n)$ at node n if MIN is of type i. In the following, we abuse the notation and use the symbol ω_- for both a behaviour strategy of MIN from Σ_-^{O}, and the transition probabilities under the strategy ω_- of the form $\omega_-(n, i, n')$.

7.3.3 Opponent-model search

In this subsection, we consider situations where MAX is certain that MIN only picks strategies described by a set of OMs Σ_-^{O} known by MAX. This assumption will be relaxed in the next subsection.

Belief States

With the knowledge of OMs, MAX can gain information during the game about the actual strategy employed by MIN and about MIN's type, using Bayesian reasoning. In order to model this, we define the following notion.

Definition 7.3.1 (Non-normalised belief state)**.** *Let $\omega_- \in \Sigma_-^{\mathrm{O}}$ be an OM. The* non-normalised belief state *(NBS) at node n about OM ω_-, written as* $\mathrm{NBS}(n, \omega_-) \in [0, 1]^t$*, is defined recursively in a top-down manner by* $\mathrm{NBS}(r, \omega_-) \coloneqq \vec{\rho}$ *and for $n' \in \mathrm{C}(n)$, if n is a node of MAX then* $\mathrm{NBS}(n', \omega_-) \coloneqq \mathrm{NBS}(n, \omega_-)$*, otherwise*[3]

$$\mathrm{NBS}(n', \omega_-)_i \coloneqq \mathrm{NBS}(n, \omega_-)_i \times \omega_-(n, i, n').$$

Let us emphasise that we define $\mathrm{NBS}(n, \omega_-)$ to be *non-normalised*; normalising it (dividing each component by the sum of all components) yields the conditional probability of MIN's types, given that n is reached and MIN plays ω_-.

Single OM

When there is only one OM ω_-, MAX has complete knowledge of MIN's strategy. Then the game becomes a single-player game with perfect information (Koller and Megiddo, 1992, Section 3.3), and the pure/mixed maxmin value reads

$$\underline{v}_+ \coloneqq \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \mathcal{U}(s_+, \omega_-) = \max_{\sigma_+ \in \Sigma_+^{\mathrm{M}}} \mathcal{U}(\sigma_+, \omega_-),$$

where the last equality is due to the linearity of $\mathcal{U}$.

This value can be computed by a bottom-up (i.e. depth-first) procedure, which recursively computes MAX's best strategies at each of their decision node. More precisely, MIN's decision nodes become chance nodes. Hence, even though MAX does not know MIN's type, they can pick actions to maximise their expected payoff with respect to MIN's type.

Example. *Consider again the game in Figure 5.6 (page 92), with ω_- as follows: MIN plays* a *if of type* 1 *or* 2*,* b *if of type* 4 *or* 5*, and* $\frac{1}{2}$a + $\frac{1}{2}$b *if of type* 3*. Given ω_- and the uniform prior $\vec{\rho}$, MAX can compute the NBS* $(\frac{1}{5}, \frac{1}{5}, \frac{1}{10}, 0, 0)$ *at node A. Normalising this NBS yields* $(\frac{2}{5}, \frac{2}{5}, \frac{1}{5}, 0, 0)$*, which means if A is reached, then the conditional probability*

[3] Recall that the index i designates the i-th component of a vector.

of MIN being of type 1 *(respectively* 2*,* 3*,* 4*, and* 5*) is* $\frac{2}{5}$ *(respectively* $\frac{2}{5}$*,* $\frac{1}{5}$*,* 0*, and* 0*). Given this NBS, action* l *yields a higher (non-normalised) payoff of* $1/2$ *than* r *(with a payoff of* 0*) at B.*

Similarly, the NBS at B is $(0, 0, \frac{1}{10}, \frac{1}{5}, \frac{1}{5})$ *and prescribes action* R *(with a payoff of* $1/2$*). At node r, MAX's payoff is simply the sum of their (non-normalised) payoff at A and B, which yields* 1*. One can check that* 1 *is indeed the best MAX can get when playing against MIN under this particular OM, and this payoff is obtained by the strategy* (l, R)*, which gives MAX a payoff of* 1 *independent of MIN's actual type.*

Proposition 7.3.2. *Let* $\langle T, \mathrm{P}, t, \vec{u}, \vec{\rho}\rangle$ *be a vector game with root r, and let* ω_- *be an OM. For* $l \in \mathrm{L}(T)$*, let* $\mathsf{eval}(l) \coloneqq \mathrm{NBS}(l, \omega_-) \cdot \vec{u}(l)$*. Then* $\texttt{MiniMax}(\mathbb{R}, \mathsf{eval}, \max, +)$ *satisfies*

$$\underline{v}_+ \coloneqq \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \mathcal{U}(s_+, \omega_-) = \max_{\sigma_+ \in \Sigma_+^{\mathrm{M}}} \mathcal{U}(\sigma_+, \omega_-) = \mathsf{val}(r),$$

and runs in polynomial time (more precisely, in $O(t|T|)$ *time).*[4]

This algorithm can be considered as a generalisation of OM search as proposed by Iida et al. (1993), which only considers games with perfect information for which OMs are described by MIN's evaluation functions.

Probabilistic OMs

We now consider the case in which MAX has the knowledge of several OMs $\omega_-^1, \ldots, \omega_-^m$ of MIN, and a probability distribution $\vec{p} = (p_1, \ldots, p_m)$ over them: MIN plays the strategy ω_-^1 with probability p_1, ω_-^2 with probability p_2, etc. In particular, the pure/mixed maxmin value is given by

$$\underline{v}_+ \coloneqq \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \sum_{j=1}^{m} p_j \mathcal{U}(s_+, \omega_-^j) = \max_{\sigma_+ \in \Sigma_+^{\mathrm{M}}} \sum_{j=1}^{m} p_j \mathcal{U}(\sigma_+, \omega_-^j).$$

This setting is not much different from the previous one, due to the linearity of u: these OMs can be merged into one single OM describing the mixed strategy $\omega_- \coloneqq p_1\omega_-^1 + \cdots + p_m\omega_-^m$.[5] In principle, one can traverse the game tree once and compute the behaviour strategy corresponding to ω_-, then run the single-OM algorithm from Proposition 7.3.2. However, with the help of NBS, one can avoid explicitly computing and storing the strategy ω_-.

Proposition 7.3.3. *Let* $\langle T, \mathrm{P}, t, \vec{u}, \vec{\rho}\rangle$ *be a vector game with root r, and let* $\omega_-^1, \ldots, \omega_-^m$ *be OMs distributed according to* $\vec{p} = (p_1, \ldots, p_m)$*. Let* $\mathsf{eval}(l) \coloneqq \sum_{j=1}^m p_j \,\mathrm{NBS}(l, \omega_-^j) \cdot \vec{u}(l)$ *for* $l \in \mathrm{L}(T)$*. Then* $\texttt{MiniMax}(\mathbb{R}, \mathsf{eval}, \max, +)$ *satisfies* $\underline{v}_+ = \mathsf{val}(r)$ *and runs in polynomial time (more precisely, in* $O(mt|T|)$ *time).*

Lexicographic OMs

Consider now the case in which MAX holds a lexicographic belief over MIN's OMs $\omega_-^1, \ldots, \omega_-^m$: MAX deems that MIN most probably follows ω_-^1; otherwise, with an

[4] The proof of this and other results in this chapter can be found in the appendix.

[5] We abuse the notation by writing ω_-^i for both the given behaviour strategy and its equivalent mixed strategy.

infinitesimally smaller probability, MIN follows ω_-^2; etc. We define the pure/mixed maxmin value in this case to be the vector of length m

$$\begin{aligned}\overrightarrow{v_+} &\coloneqq \operatorname*{lexmax}_{s_+ \in \Sigma_+^{\mathrm{P}}} \left(\mathcal{U}(s_+, \omega_-^1), \ldots, \mathcal{U}(s_+, \omega_-^m)\right) \\ &= \operatorname*{lexmax}_{\sigma_+ \in \Sigma_+^{\mathrm{M}}} \left(\mathcal{U}(\sigma_+, \omega_-^1), \ldots, \mathcal{U}(\sigma_+, \omega_-^m)\right) \in \mathbb{R}^m,\end{aligned}$$

where lexmax is lexicographic maximum over vectors of length m. In other words, if there is a unique optimal strategy against ω_-^1, then this strategy is chosen; otherwise, ties are broken according to their values against ω_-^2, and so on.

This setting can be regarded as an instance of probabilistic OMs, where the distribution over OMs is $\overrightarrow{p_\varepsilon} = (1, \varepsilon, \varepsilon^2, \ldots, \varepsilon^{m-1})$ with ε an indeterminate interpreted as an infinitesimally small value. However, we also give a direct algorithm below. We write $\mathrm{NBSM}(n)$ for the $m \times t$ NBS matrix

$$\left(\mathrm{NBS}(n, \omega_-^1)^\intercal, \ldots, \mathrm{NBS}(n, \omega_-^m)^\intercal\right),$$

and $+^m$ for component-wise addition of vectors in $\mathbb{R}^m$.

Proposition 7.3.4. *Let $\langle T, \mathrm{P}, t, \vec{u}, \vec{\rho}\rangle$ be a vector game with root r, and let $\omega_-^1, \ldots, \omega_-^m$ be OMs with a lexicographic interpretation. Let* $\mathsf{eval}(l) \coloneqq \mathrm{NBSM}(l) \times \vec{u}(l) \in \mathbb{R}^m$ *for $l \in \mathrm{L}(T)$. Then* $\texttt{MiniMax}(\mathbb{R}^m, \mathsf{eval}, \mathrm{lexmax}, +^m)$ *satisfies* $\overrightarrow{v_+} = \mathsf{val}(r)$ *and runs in polynomial time (more precisely, in $O(mt|T|)$ time).*

Nondeterministic OMs

The last case we consider is when MAX has no probability distribution over MIN's OMs: MIN's strategy is only known to be among $\omega_-^1, \ldots, \omega_-^m$. This situation is similar to planning under adversarial cost functions (McMahan et al., 2003). The maxmin value is then

$$\underline{v}_+ \coloneqq \max_{\varsigma_+ \in \Sigma_+} \min_{1 \le j \le m} \mathcal{U}(\varsigma_+, \omega_-^j),$$

which, in general, is different depending on whether Σ_+ is Σ_+^{P} or Σ_+^{M}. MIN now has (*a priori*) more agency than in the case of probabilistic OMs, since they can choose from a larger (but still limited) set of strategies.

We first consider pure maxmin. For $f, g \in \mathcal{P}_{<\infty}(\mathbb{R}^m)$, define the following operator:

$$f \oplus^m g \coloneqq \{(v_j + v'_j)_{1 \le j \le m} \mid \vec{v} \in f, \vec{v}\,' \in g\} \subseteq \mathbb{R}^m.$$

$\oplus^m$ is similar to $\sqcap$, but takes component-wise sum instead of component-wise minimum between all pairs of vectors.

Proposition 7.3.5. *Let $\langle T, \mathrm{P}, t, \vec{u}, \vec{\rho}\rangle$ be a game with root r, and let $\omega_-^1, \ldots, \omega_-^m$ be OMs with a nondeterministic interpretation. For $l \in \mathrm{L}(T)$, let* $\mathsf{eval}(l) \coloneqq \{\mathrm{NBSM}(l) \times \vec{u}(l)\}$. *Then* $\texttt{MiniMax}(\mathcal{P}_{<\infty}(\mathbb{R}^m), \mathsf{eval}, \cup, \oplus^m)$ *satisfies*

$$\underline{v}_+ \coloneqq \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \min_{1 \le j \le m} \mathcal{U}(s_+, \omega_-^j) = \max_{\vec{v} \in \mathsf{val}(r)} \min_{1 \le j \le m} v_j.$$

This algorithm is exponential time in the worst case, which is not surprising since we have already seen that computing the pure maxmin value against only two nondeterministic OMs is already NP-hard, even if MAX has complete information (cf. Proposition 4.4.8, page 58).

Compared to Proposition 5.4.2 (page 97), the knowledge of OMs transforms MAX's incomplete information about MIN's *type* into their incomplete information about MIN's *strategy*. Situational values are now sets of vectors of length m (instead of t). Each such vector represents a strategy of MAX by its expected payoff against each OM. However, in contrast with probabilistic OMs, we cannot collapse each vector to a real number, since we have no distribution over the OMs. Still, reduction by weak dominance can be used just as for pure maxmin without any opponent model (cf. Subsection 6.3.1).

From another perspective, this algorithm computes the normal form of the game restricted to MIN's fixed m strategies, which intuitively justifies the correctness of Proposition 7.3.5 for pure maxmin.

As for mixed maxmin, one can modify the separation oracle in the LP algorithm of Koller and Megiddo (1992): now the oracle only computes MIN's best responses from the m OMs. This yields a polynomial-time algorithm.

7.3.4 Opponent models with uncertainty

We now come to our second type of opponent-model search, which is about the case in which a set of OMs of MIN is available, but MAX is not certain that MIN will behave as one of them. We focus on the case in which there is only one single OM ω_-, which encompasses as well the case of several OMs with a probability distribution or lexicographic interpretation, as discussed in the last subsection.

We assume that with probability p^∞, which is known to MAX, MIN does not follow ω_-, in which case their behaviour is arbitrary and unpredictable, i.e. they can play any $s_- \in \Sigma_-^{\mathrm{P}}$; and with probability $1 - p^\infty$, MIN follows ω_-. Intuitively, p^∞ quantifies MAX's uncertainty about MIN's behaviour. This may arise, for instance, when the OMs are given by an estimate of MIN's gameplay level: if MAX correctly estimates MIN's level, then MAX has a pretty good idea of what MIN will play as strategy; otherwise MAX cannot predict MIN's behaviour at all.

Formally, we define the following maxmin value:

$$\underline{v}_+ \coloneqq \max_{\varsigma_+ \in \Sigma_+} \big((1 - p^\infty)\mathcal{U}(\varsigma_+, \omega_-) + p^\infty \min_{s_- \in \Sigma_-^{\mathrm{P}}} \mathcal{U}(\varsigma_+, s_-)\big),$$

where Σ_+ is either Σ_+^{P} or Σ_+^{M}.

Example. *Consider again the vector game in Figure 5.6 (page 92) and the OM ω_- "MIN plays* a *if of type* 1 *or* 2, b *if of type* 4 *or* 5, *and* $\frac{1}{2}$a + $\frac{1}{2}$b *if of type* 3*". The best strategy of MAX against ω_- is* (l, R) *with a payoff of* 1. *However, this strategy does not fare so well if MIN's strategy is not ω_- (or when p^∞ is close to* 1*): in the worst case, MIN plays* b *if of type* 1 *or* 2, *and* a *if of type* 4 *or* 5. *Against this strategy, MAX's expected payoff from playing* (l, R) *is only* 1/5. *On the other hand, the pure maxmin strategy* (l, L) *only has an expected payoff of* 1/2 *against ω_-, and so does the mixed maxmin strategy (which is the uniform strategy); hence, neither is optimal when p^∞ is close to* 0.

It is clear from this example that the maxmin value and the optimal strategies depend on the value of p^∞. This demonstrates a conflict between robustness and performance: MAX desires to be cautious and robust against MIN's unpredictable behaviour occurring with probability p^∞, and at the same time to improve their performance by exploiting their knowledge of the OM, which correctly predicts MIN's strategy with probability $1 - p^\infty$.

We now show how to modify algorithms from the last sections to compute the maxmin value.

Mixed maxmin

For the mixed maxmin value, we can use the LP algorithm by Koller and Megiddo (1992) with a minor modification of the separation oracle: given a threshold v and a mixed strategy σ_+ for MAX, the separation oracle should now, apart from computing MAX's payoff v_{BR} with σ_+ against MIN's best response, also compute MAX's payoff against the OM $v_{\mathsf{OM}} = \mathcal{U}(\sigma_+, \omega_-)$, then check whether $(1 - p^\infty)v_{\mathsf{OM}} + p^\infty v_{\mathsf{BR}} \geq v$ holds.

Example. *In the game in Figure 5.6 (page 92) with ω_- as above, one can use this algorithm to verify that MAX's optimal strategy is* (l, R) *for $p^\infty \leq 5/8$, otherwise it is the uniform strategy. This confirms that when nondeterministic behaviour happens with a small enough probability, it is worth deviating from maxmin strategies in order to exploit the OM.*

Pure maxmin

For pure strategies, we build on the algorithm for a single OM (Proposition 7.3.2) and Ginsberg's algorithm. To cope with non-locality (due to MIN's partially unpredictable behaviour), we use situational values that are finite sets of ordered pairs $\langle s, \vec{v}\rangle$, with $s \in \mathbb{R}$ and $\vec{v} \in \mathbb{R}^t$. We call such a pair an *annotated vector*; it implicitly represents a strategy for MAX for which the payoff against ω_- is s, and the worst payoff against MIN's unpredictable behaviour is given by $\vec{v}$.

We write $\mathcal{P}_{<\infty}(\mathbb{R} \times \mathbb{R}^t)$ for the set of all finite sets of annotated vectors, and for $f, g \in \mathcal{P}_{<\infty}(\mathbb{R} \times \mathbb{R}^t)$, we define $f \oplus^{1,t} g \subseteq \mathbb{R} \times \mathbb{R}^t$ to be the set

$$\{\langle s + s', (\min(v_i, v_i'))_{1 \leq i \leq t}\rangle \mid \langle s, \vec{v}\rangle \in f, \langle s', \vec{v}\,'\rangle \in g\}.$$

Proposition 7.3.6. *Let $\langle T, \mathrm{P}, t, \vec{u}, \vec{\rho}\rangle$ be a vector game with root r, ω_- be an OM, and $p^\infty \in [0, 1]$ a probability that MIN does not follow ω_-. For all $l \in \mathrm{L}(T)$, let*

$$\mathsf{eval}(l) \coloneqq \{\langle \mathrm{NBS}(l, \omega_-) \cdot \vec{u}(l), \vec{u}(l)\rangle\} \in \mathcal{P}_{<\infty}(\mathbb{R} \times \mathbb{R}^t).$$

Then $\texttt{MiniMax}(\mathcal{P}_{<\infty}(\mathbb{R} \times \mathbb{R}^t), \mathsf{eval}, \cup, \oplus^{1,t})$ *satisfies*

$$\underline{v}_+ = \max_{\langle s, \vec{v}\rangle \in \mathsf{eval}(r)} \big((1 - p^\infty)s + p^\infty(\vec{q} \cdot \vec{v})\big).$$

Notice that when combining two annotated vectors at a MIN's node, the scalar part is additive; this reflects the fact that when following the (single) OM, MIN has no agency, just as in the case without uncertainty.

Example. *Using the algorithm above for the game in Figure 5.6 (page 92) with the aforementioned OM ω_-, we find out that MAX's optimal strategy is* (l, R) *for $p^\infty \leq 5/7$, otherwise* (l, L) *or* (r, R). *Again, this indicates that it may be worth deviating from maxmin strategies so as to exploit an OM.*

Note that Proposition 7.3.6 implies Proposition 5.4.2 (page 97), since the algorithm in Proposition 7.3.6 generalises Ginsberg's algorithm in Proposition 5.4.2, which corresponds to the special case $p^\infty = 1$. In particular, this means Proposition 7.3.6 is not polynomial time.

Interestingly, the case $p^\infty \neq 1$ supports more sound pruning.[6] Indeed, let $n \in \mathrm{N}(T)$ and $\langle s, \vec{v}\rangle, \langle s', \vec{v}\,'\rangle \in \mathsf{val}(n)$. Pruning $\langle s', \vec{v}\,'\rangle$ because of $\langle s, \vec{v}\rangle$ is sound if MAX is never worse-off in the game by choosing $\langle s, \vec{v}\rangle$ instead of $\langle s', \vec{v}\,'\rangle$ at n. Now since scalar parts are summed up, if $s > s'$ holds, then $\langle s, \vec{v}\rangle$ has an advantage $s - s'$ over $\langle s', \vec{v}\,'\rangle$; contrastingly, for the vectorial part, components for which $\vec{v}$ is larger than $\vec{v}\,'$ might be erased by the combination (via component-wise min) of vectors at an ancestor of n, so that the advantage of $\vec{v}$ can be annihilated at the root. Hence, in the worst case, $\vec{v}\,'$ can keep all advantages it has compared to $\vec{v}$, while $\vec{v}$ can lose all its advantages.

To summarise, we can prune $\langle s', \vec{v}\,'\rangle$ when it holds that

$$(1 - p^\infty)(s - s') \geq p^\infty \sum_{1 \leq i \leq t} \big(q_i \max(v'_i - v_i, 0)\big),$$

which indeed generalises the pruning condition related to dominance in Subsection 6.3.1: when $p^\infty = 1$, the inequality above reduces to $v'_i \leq v_i$ for all i, which means $\vec{v}$ dominates $\vec{v}\,'$.

7.4 Recursive opponent modelling

We now propose an application of the algorithms presented in the last section to the computation of optimal strategies with recursive opponent models. We formulate a quite general setting, where various types of opponent models naturally arise.

Limitations of the best-defence model

In general, in a game with incomplete information, both players have incomplete information, rather than just MAX. As a result, the best-defence model usually gives MIN too much power.

Example. *Consider the game in Figure 7.3, where MAX has* 3 *types (with a uniform prior) and MIN has only* 1*. Then MIN has incomplete information. If MAX reasons according to the best-defence model, then both actions* a *and* b *have a value of* 0*: MAX of type i reasons that MIN will play* a_i *at node A, and* b_j *at node B for some* $j \neq i$*. The culprit is that MAX assumes MIN is aware of MAX's type so that MIN can adapt their strategy to MAX's type. However, if MAX realises MIN is unaware of their type, then MAX will prefer* a *since under uniform common prior over MAX's types,* a *yields an expected payoff of* $2/3$*, compared to* b*'s* $1/3$*.*

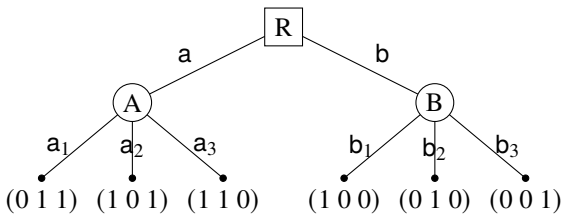


Figure 7.3: A vector game with 3 possible types of MAX.

[6] Recall that for Ginsberg's algorithm, the only sound pruning is the elimination of dominated strategies (cf. Subsection 6.3.1).

On the other hand, computing maxmin strategies for the original game tree without using the best-defence model is not ideal either, for these strategies fail to exploit any assumption one may have about their adversary, such as that they have limited computational power or reasoning depth, or that they have a predictable behaviour. Such assumptions make sense in particular when playing against humans (Iida et al., 1993; Stahl and Wilson, 1995; Dhami, 2019).

Proposed framework

We propose a framework that can be considered as a generalisation of the cognitive hierarchy model (Camerer et al., 2004), and as a counterpart of interactive POMDPs (Doshi et al., 2020) for competitive games. The idea is to define *level-k* strategies to be the optimal strategies against an adversary of level $k-1$, and recursively down to level 0. Our framework serves as a compromise between the best-defence model and the full game, and can be used to find better strategies against non-omnipotent and non-omniscient players; in particular, it generalises the best-defence model. Moreover, it can be used to explain real-life human psychological gameplay in games such as Bridge, as we illustrate at the end of this section.

We give a parametrizable definition of (1) how level-0 strategies are defined, (2) how optimal strategies at a given level are aggregated, and (3) how strategies of various levels are aggregated. Let Σ_+^0, Σ_-^0 be non-empty sets of strategies of MAX and MIN. Moreover, for $i \in \{+,-\}$, let $\oplus_i : \mathcal{P}(\Sigma_i) \to \Sigma_i$ map any set of (pure or mixed) strategies of player i to a single strategy of player i, and $\mathrm{BR}_i : \Sigma_{-i}^* \to \mathcal{P}(\Sigma_i)$ map any tuple of strategies of player $-i$ to a set of strategies of player i. $\oplus$ will aggregate strategies of a player at a given level, and BR will compute the set of optimal strategies given a tuple of opponent models (one per lower level).[7]

Definition 7.4.1 (level-k strategies)**.** *Let Σ_i^0, $\oplus_i$, and BR_i be defined as above for all $i \in \{+,-\}$. The set of* level-0 *strategies for player i is defined to be Σ_i^0. For $k \geq 1$, the set of* level-k strategies *for player i, denoted by Σ_i^k, is defined to be $\mathrm{BR}_i\left(\oplus_{-i}(\Sigma_{-i}^{k-1}), \oplus_{-i}(\Sigma_{-i}^{k-2}), \ldots, \oplus_{-i}(\Sigma_{-i}^0)\right)$.*

In short, the level-k strategies of player i are the best responses (computed by BR_i, the *best-response function*) against an opponent using the strategy $\oplus_{-i}(\Sigma_{-i}^{k'})$ (computed by $\oplus_{-i}$, the *intra-level aggregation*) at each level k' for $0 \leq k' < k$. The boundary conditions, i.e. the level-0 strategies, are given by Σ_i^0, which can come from maxmin strategies under the best-defence model, randomly chosen strategies (McMahan et al., 2003), modelling assumptions for human players (Wright and Leyton-Brown, 2019), etc.

Example. *The Poisson-CH model in Camerer et al. (2004) is captured by choosing Σ_+^0 and Σ_-^0 to be the set of all pure strategies of MAX and MIN, the intra-level aggregation $\oplus$ to map any set of strategies to the uniform mixture of the set, and the best-response function* BR *to map a tuple of strategies $(\sigma_{-i}^{k-1}, \ldots, \sigma_{-i}^0)$ to the set of all pure best responses to the mixed strategy $p_{k-1}\sigma_{-i}^{k-1} + \cdots + p_0\sigma_{-i}^0$, where $p_{k-1}, \ldots, p_0$ follow some Poisson distribution.*

Although we do not show it here, this framework is also general enough to encompass many iterative approaches of solving games: iterative best response (also called best

[7] The framework could be easily adapted to more general functions, e.g. an aggregation of the strategies at the same level into a set or a tuple of strategies. It could also be easily applied to general games, in normal form or extensive form, beyond the two-player and zero-sum assumptions.

response dynamics); fictitious play (Brown, 1951; Cloud et al., 2023); double oracle (McMahan et al., 2003); max^n or prob-max^n (Sturtevant et al., 2006); etc.

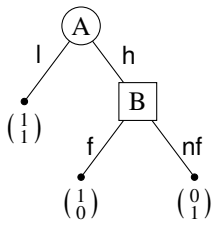
An interesting choice for intra-level aggregation $\oplus_i : \mathcal{P}(\Sigma_i) \to \Sigma_i$ for all i is the uniform mixture, as in the previous example. With this, many situations can be modelled by using different best-response functions BR for inter-level aggregation, in particular using the algorithms presented in previous sections.

- Using the probabilistic OM search algorithm in Proposition 7.3.3, we can model situations where each player i at level k has a subjective distribution (described by $p_{i,k}^{k-1}, p_{i,k}^{k-2}, \ldots, p_{i,k}^{0}$) over player $-i$'s reasoning levels, obtained for instance by fitting a model against a population of possible opponents;
- Setting $p_{i,k}^{k-1} = 1$ for all i and k in the previous model, we can model the situation where each player at level k assumes their opponent reasons at level exactly $k-1$;
- Using the lexicographic OM search algorithm in Proposition 7.3.4, we can model situations where player i at level k assumes player $-i$ to reason at level $k-1$, tie-breaks equivalent strategies by assuming them to reason at level $k-2$, and so on;
- Using the nondeterministic OM search algorithm in Proposition 7.3.5, we can model situations where player i at level k assumes player $-i$ to reason at an unknown level lower than k (then the incomplete information about player $-i$'s type becomes one about their level, which is, in general, much smaller);
- Using OM search algorithm with uncertainty in for example Proposition 7.3.6, we can model situations where player i assumes that with a certain probability $-i$ does not reason at any level lower than k.

Let us also emphasise that the straightforward generalisation of the framework in Definition 7.4.1 to general games allows, for instance, to take into account one's partner's incomplete information in multiplayer games, akin to interactive POMDPs.

A real-life example

We now give an example application of our formalism, which captures the psychological strategies of a contract Bridge deal played in a Bridge tournament. We present the abstract version of the game in Figure 7.4 (left); for the Bridge deal itself, see Deal 3 (page 169) in Appendix A.4.1.



k	MAX	MIN 1	MIN 2
0	{nf}	{l, h}	{l, h}
1	{nf}	{h}	{l, h}
2	{f}	{h}	{l, h}
3	{f}	{h}	{h}

Figure 7.4: Level-k strategies in a vector game with 2 types of MIN.

In this game, the common prior about MIN's types is given by $p_1 = 0.4$ and $p_2 = 0.6$. For the recursive reasoning, for $i \in \{+, -\}$, $\oplus_i$ is given by the uniform

mixture, BR_i is given by the lexicographic model, and the level-0 strategies for both players are their pure maxmin strategies. Figure 7.4 (right) shows the level-k strategies for MAX, MIN if of type 1, and MIN if of type 2, and for $k = 0, \ldots, 3$. For instance, if MIN is of type 2, then their level-1 strategies are l and h.

The first few levels of the recursive reasoning proceed as in Figure 7.4 (right). In the following, we write $\sigma^1_- | \sigma^2_-$ for MIN's strategy to play σ^1_- if of type 1 and to play σ^2_- if of type 2; and $\frac{\mathsf{l}+\mathsf{h}}{2}$ for the uniform mixed strategy $\frac{1}{2}\mathsf{l} + \frac{1}{2}\mathsf{h}$.

k = 0: MAX prefers nf, which achieves a maxmin value of 0.6, against 0.4 for f; both types of MIN are indifferent between l and h since both yield a minmax value of 1.

k = 1: Against $(\oplus(\Sigma^0_-)) = (\frac{\mathsf{l}+\mathsf{h}}{2} | \frac{\mathsf{l}+\mathsf{h}}{2})$, MAX's best strategy is still nf; however, against $(\oplus(\Sigma^0_+)) = (\mathsf{nf})$, type-1 MIN prefers h which yields a value of 0.

k = 2: Against $(\oplus(\Sigma^1_-), \oplus(\Sigma^0_-))$, MAX now prefers f, which is strictly better than nf against $\oplus(\Sigma^1_-) = \mathsf{h} | \frac{\mathsf{l}+\mathsf{h}}{2}$ since the NBS of MAX at node B judges MIN is more likely to be of type 1 than of type 2 if MIN plays $\mathsf{h} | \frac{\mathsf{l}+\mathsf{h}}{2}$;

k = 3: At level-3, type-1 MIN still prefers h: h and l are equivalent against $\oplus(\Sigma^2_+) = \mathsf{f}$, but h is preferred against $\oplus(\Sigma^1_+) = \mathsf{nf}$; but now type-2 MIN also prefers h!

As it turns out, this recursive reasoning captures perfectly what happened during the Bridge deal, where MAX was at level 2 and therefore chose f (rather than the maxmin strategy nf) while MIN, being of type 2, reasoned at level 3 and used strategy h to defeat MAX. Another interesting point is that level-k strategies are not necessarily weakly dominant; hence they incorporate some notion of risk. For instance, MAX's level-2 strategy f performs better than the maxmin strategy nf against MIN of level 1 (it pays 0.7 instead of 0.6), but it performs worse against MIN of level 3 (0.4).

7.5 Conclusion

In this chapter, we have first argued about the need to go beyond the best-defence model in general games with incomplete information. Several promising ideas to this end have been briefly discussed. One of them, opponent-model search, is thoroughly explored from both modelling and algorithmic perspectives. Near the end, we have presented a general framework of recursive opponent modelling, instantiations of which are widely used in the literature to compute (exactly or approximately) different solution concepts.

Prospectives

As we have said in the introduction of this chapter, most ideas in this chapter are still in their infancy. Hence, a major axis of future work will be the ironing out of the details. To cite a few directions:

- exploring algorithms such as counterfactual regret minimisation to compute the approximate values of behaviour maxmin of CGIIs limited to a smaller size of the universe;
- considering opponent-model search for compactly represented games, for opponent models not given as oracles of behaviour strategies of MIN, for other kinds of uncertainty over the opponent models;

- exploring other recursive modellings in the literature, especially those in the field of behavioural game theory, and showing how they are captured by our general framework of recursive opponent modelling;
- giving an epistemic foundation (e.g. by using doxastic or epistemic logic) of our recursive framework, in the spirit of the notion of rationalisability in epistemic game theory.

Conclusion and perspectives

In this dissertation, we have focused on game theory, an important field relevant to decision-making and artificial intelligence, and at the intersection of economics, mathematics, and computer science. Among the manifold topics in game theory, we have chosen to focus on finding robust strategies for extensive-form games (EFGs), which are models for sequential multi-agent interaction, due to the motivation and inspiration from the game of Bridge.

After surveying vast and interconnected subjects related to our work and presenting the rudiments of game theory, we have proceeded to study the computational complexity of computing the pure maxmin value for two-team EFGs of chance or of no chance, and with different degrees of imperfect information for team MAX and team MIN. In this thorough study, we have proposed new polynomial-time algorithms; strengthened existent hardness results in the literature by showing that they hold even under strong restrictions (Boolean payoffs, at most 2 agents per team, etc); proved new complexity results. Our work thus provides a complete complexity landscape for pure maxmin. Then we have further advanced on three different axes: the complexity of compactly represented games; the complexity of games with a known and finite set of strategies for the opponent (i.e. opponent models); and the complexity of finding an upper bound or the exact value of pure maxmin. We have carefully chosen the reductions we use to (1) give better intuitions about how EFGs can be used to encode difficult problems (e.g. NP-hard); (2) show that the hardness results hold under minimal assumptions; (3) facilitate future work on inapproximability or parameterised complexity results.

Then, we have turned our attention to games with incomplete information, which are special cases of games with imperfect information that describe situations in which players do not have common knowledge about the game they play. After discussing the notion of incomplete information (and contrasting it with imperfect information), we have introduced a new formalism called combinatorial games with incomplete information (CGIIs). CGIIs are games with incomplete information that have no chance node except for the initial drawing by Nature, and only public actions. CGII is a formalism that (1) minimally generalises the formalism of combinatorial games to allow incomplete information; (2) minimally captures the essence of the notion of incomplete information; (3) allows modelling the card play of Bridge. We have started the study of CGIIs by showing that finding optimal strategies in CGIIs has the same complexity as in the more general model of EFGs, which implies that CGIIs and EFGs are equally expressive, and warrants interest in studying this new formalism as a simpler and more minimal way to emphasise public actions and incomplete information in sequential games. In proving the hardness results for CGIIs, we make use of various gadgets of reduction to show how notions such as concurrent actions can be encoded in CGIIs.

Next, we have switched the focus to algorithmic aspects of solving CGIIs. The best-defence model from the literature has been introduced and firmly justified. Notions

of strategy fusion and non-locality have been explained in details, to make them more intuitive, so as to understand where the difficulty of solving games with incomplete information comes from. A depth-first algorithm called Ginsberg's algorithm, which computes the exact pure maxmin value of CGIIs under the best-defence model, has been presented in its fully general form. Then we have studied various sources of exact optimisations of Ginsberg's algorithm, such as strategy prunings (e.g. elimination of dominated strategies) and game tree prunings (e.g. alpha-beta search). These optimisations, although being designed for Ginsberg's algorithm or inspired by human gameplay strategies in Bridge, have been presented in an entirely general way, to emphasise that they are applicable beyond Ginsberg's algorithm and CGIIs.

Finally, we have briefly discussed reasoning in games, and more specifically in CGIIs. The interest of reasoning about opponents' strategies and state of mind lies in both the need to go beyond the best-defence model (which greatly reduces the complexity, but can be suboptimal when the opponent has too much incomplete information), and the need to take into account the fact that opponents (especially when they are humans) are unlikely to play perfectly against our interest, for various reasons, e.g. limitations of computational resources. To show the algorithmic feasibility of reasoning about opponents' strategies, we have designed algorithms for opponent-model search for CGIIs. These algorithms can be used in a very general framework of recursive opponent modelling, which encompasses many solution concepts or iterative algorithms for solution concepts, especially from the fields of behavioural game theory and epistemic game theory.

Perspectives

To summarise, we have looked at games with incomplete information from three different perspectives: complexity, algorithmics, and reasoning.

For complexity, we may tie up some loose ends, e.g. the precise relationship between the oracle games that we have defined and GDL, the complexity of games with absentmindedness (i.e. imperfect information but not multi-agent perfect recall), etc. It will also be desirable to refine the proofs of hardness results for CGII to make them simpler to understand, or easier to be adapted to other variants of the decision problem PURE MAXMIN.

For algorithmics, the most interesting directions are to look for exact (and ideally depth-first) algorithms for CGIIs that are not under the best-defence model; to incorporate Monte Carlo techniques into Ginsberg's algorithm; and to test different optimisations on well-founded benchmarks.

Finally, for reasoning in games, the aspect the least studied by us, there are even more fascinating topics to investigate, including, but not restricted to, formal definitions for various kinds of recursive reasoning in games (about knowledge, opponents' strategies or state of mind, etc.); algorithms for automating such reasoning; epistemic characterisations for recursive modelling; etc.

Appendices

Appendix A

Reminders of definitions

In this chapter, we collect reminders of notions from different fields that are used in the main text, except those from game theory, which have already been given in Chapter 3.

A.1 Graph theory

Definition A.1.1 (Directed graph)**.** *A* (directed) graph *is a pair* (V, E)*, where* V *is a finite set (the elements of which are called* vertices*), and* $E \subseteq V \times V$ *is a finite set of pairs of vertices (called* edges*).*

Remark. *In this dissertation, we will use the terms* vertex *and* node *almost interchangeably. With "vertex", we emphasise that it is an element in the graph; with "node", we emphasise that it is a place where a player makes a decision (i.e. the notion of decision node).*

Definition A.1.2 (Path)**.** *Let* $v, v' \in V$ *be two vertices in a graph. A* path *from* v *to* v' *is a finite (possibly empty) sequence of edges* $(v, v_1), (v_1, v_2), \ldots, (v_{n-1}, v_n), (v_n, v') \in E$*, also denoted by* $(v, v_1, \ldots, v_n, v')$*. Two paths are considered identical if they are composed of exactly the same sequence of edges.*

Definition A.1.3 (DAG)**.** *A* directed acyclic graph *is a directed graph such that there is no non-empty path from any vertex to itself.*[1]

Definition A.1.4 (Tree)**.** *A* tree *is a DAG* (V, E) *such that there is a unique vertex* $r \in V$ *(called the* root*), such that for every* $v \in V$*, there is a unique path from* r *to* v*.*

We denote an arbitrary tree by a triple $T = (V, E, r)$.

Definition A.1.5 (Parent and child)**.** *Let* (V, E, r) *be a tree. Let* $v, v' \in V$*. If* $(v, v') \in E$*, we call* v' *a* successor *or a* child *of* v*, and* v *the* predecessor *or the* parent[2] *of* v'*. If there is a non-empty path from* v *to* v'*, we call* v' *a* descendant *of* v*, and* v *an* ancestor *of* v'*. A vertex without successor is called a* leaf*. A vertex that is not a leaf is called* non-leaf *or* internal *vertex.*

For a tree T, we write $\mathrm{N}(T)$ and $\mathrm{L}(T)$ for the set of its nodes and its leaves, respectively. For an internal vertex $v \in \mathrm{N}(T) \setminus \mathrm{L}(T)$, we write $\mathrm{C}(v)$ for the set of children of v.

A.2 Complexity classes

The complexity classes we use in this dissertation have the same standard meaning as in the literature. Readers can refer to Arora and Barak (2009, Definition 5.3, for instance) for formal definitions. Below, we only give an intuitive characterisation of the classes that we use.

- The class P contains all languages that can be decided by a polynomial-time Turing machine.

[1] Such a non-empty path, which contains at least one edge, is called a *cycle*, hence the name "acyclic".

[2] In a tree, every vertex except the root has a unique predecessor/parent, hence the article "the".

- The class NP contains all languages that can be decided by a polynomial-time nondeterministic Turing machine. Equivalently, NP contains all languages L such that
$$x \in L \iff \exists y \in \{0,1\}^{p(|x|)}, (x,y) \in L',$$
where p is a polynomial and L' is a language in P.

- The class coNP is the set of complements of languages in NP. Equivalently, coNP contains all languages L such that
$$x \in L \iff \forall y \in \{0,1\}^{p(|x|)}, (x,y) \in L',$$
where p is a polynomial and L' is a language in P.

- The class Δ_2^{P}, also written as P^{NP}, contains all languages that can be decided by a polynomial-time Turing machine with the help of an NP oracle.[3]

- The class Σ_2^{P}, alternatively defined as $\mathsf{NP}^{\mathsf{NP}}$, contains all languages L such that
$$x \in L \iff \exists y_1 \in \{0,1\}^{p(|x|)}, \forall y_2 \in \{0,1\}^{p(|x|)}, (x,y_1,y_2) \in L',$$
where p is a polynomial and L' is a language in P.

- The class Π_2^{P}, alternatively defined as $\mathsf{coNP}^{\mathsf{NP}}$, is the set of complements of languages in Σ_2^{P}. Equivalently, Π_2^{P} contains all languages L such that
$$x \in L \iff \forall y_1 \in \{0,1\}^{p(|x|)}, \exists y_2 \in \{0,1\}^{p(|x|)}, (x,y_1,y_2) \in L',$$
where p is a polynomial and L' is a language in P.

- The class PSPACE contains all languages that can be decided by a polynomial-space Turing machine.

- The class EXP contains all languages that can be decided by an exponential-time Turing machine.

- The class NEXP contains all languages that can be decided by an exponential-time nondeterministic Turing machine. Equivalently, NEXP contains all languages L such that
$$x \in L \iff \exists y \in \{0,1\}^{2^{p(|x|)}}, (x,y) \in L',$$
where p is a polynomial[4] and L' is a language in P.

- The class $\mathsf{NEXP}^{\mathsf{NP}}$ contains all languages that can be decided by an exponential-time nondeterministic Turing machine with the help of an NP oracle. Equivalently, $\mathsf{NEXP}^{\mathsf{NP}}$ contains all languages L such that
$$x \in L \iff \exists y_1 \in \{0,1\}^{2^{p(|x|)}}, \forall y_2 \in \{0,1\}^{2^{p(|x|)}}, (x,y_1,y_2) \in L',$$
where p is a polynomial and L' is a language in P.

[3] An NP oracle decides the membership of a language in NP in one step. However, the Turing machine making use of such an oracle still needs time to write down the queries to the oracle, which, in this case, takes exponential time.

[4] Hence, the size of the witness y is exponential in x.

We also use some less known complexity classes. D_k^P is the set of languages L that can be written as $L = L_1 \cap L_2$ with $L_1 \in \Sigma_k^\mathsf{P}$ and $L_2 \in \Pi_k^\mathsf{P}$, or equivalently as the difference of two languages in Σ_k^P (or two languages in Π_k^P). The most well-known one is D_1^P, which is also called DP in the literature. Intuitively, DP corresponds to problems that concern the optimal value of an optimisation problem in NP (Papadimitriou, 1994).

By generalising the argument in Papadimitriou (1994, Theorem 17.1), one can prove the D_k^P-hardness of a language L by showing that there are a Σ_k^P-hard language L_1, a Π_k^P-hard language L_2, and a polynomial-time reduction f such that

$$\langle x, y\rangle \in L_1 \times L_2 \iff f(\langle x, y\rangle) \in L.$$

A.3 Sets

In this dissertation, we usually write S for a finite set without a particular structure, while V for a set of vertices, a poset, or lattice (see Subsection A.3.2 for their definitions).

Definition A.3.1 (Cardinality)**.** *Let S be a finite set. The* cardinality *(or* size*) of S, denoted by $|S|$, is the number of elements in S.*

Definition A.3.2 (Powerset)**.** *Let S be a finite set. The* powerset *of S, denoted by $\mathcal{P}(S)$, is the set of all subsets of S:*

$$\mathcal{P}(S) = \{S' \mid S' \subseteq S\}.$$

Definition A.3.3 (Partition)**.** *Let S be a finite set. A* partition *of S is a set $\mathcal{S}$ of non-empty subsets of S such that the subsets in $\mathcal{S}$ are pairwise disjoint, and their union is S:*

$$\forall S', S'' \in \mathcal{S}, S' \neq S'' \implies S' \cap S'' = \emptyset,$$

and $\bigcup_{S' \in \mathcal{S}} = S$. A partition is said to be the finest *partition if it only contains singleton sets; it is said to be the* coarsest *partition if $\mathcal{S} = \{S\}$.*

Definition A.3.4 (Simplex)**.** *Let S be a countably finite set. The* simplex *of S, denoted by $\Delta(S)$, is the set of all probability distributions over S, i.e. mappings $p : S \to [0, 1]$ such that $\sum_{s \in S} p(s) = 1$.*

A.3.1 Family algebra

We present here the family algebra: algebraic conventions for binary operators over families of a finite set. This terminology is introduced by Knuth (2011).

Definition A.3.5 (Family)**.** *Let S be a finite set. A* family *of S is an element of $\mathcal{P}(\mathcal{P}(S))$, i.e. a set of subsets of S.*

Example. *A partition of S is a family of S. Here are some more simple examples of families:*

- *The* empty family $\emptyset$*, which contains no subset of S.*
- *The* unit family $\{\emptyset\}$*, which only contains the empty subset of S.*
- *The* elementary families $\big\{\{j\}\big\}$ *for $j \in S$.*

Let $f, g \in \mathcal{P}(\mathcal{P}(S))$ be two families of S. One can construct new families of S using the following operators.

union $f \cup g = \{\alpha \mid \alpha \in f \vee \alpha \in g\}$;

intersection $f \cap g = \{\alpha \mid \alpha \in f \wedge \alpha \in g\}$;

join $f \sqcup g = \{\alpha \cup \beta \mid \alpha \in f \wedge \beta \in g\}$;

meet $f \sqcap g = \{\alpha \cap \beta \mid \alpha \in f \wedge \beta \in g\}$;

maximal elements $f^{\uparrow} = \{\alpha \in f \mid \forall \beta \in f, \alpha \subseteq \beta \implies \alpha = \beta\}$;

minimal elements $f^{\downarrow} = \{\alpha \in f \mid \forall \beta \in f, \alpha \supseteq \beta \implies \alpha = \beta\}$.

In the main text, we refer to these operators as *operators from the family algebra*.

A.3.2 Lattices

The following definitions on posets and lattices are based on Davey and Priestley (2002).

Lattices as ordered sets

Lattices can be defined as partially ordered sets satisfying some additional properties.

Definition A.3.6 (Poset)**.** *Let V be a set and $\preceq$ be a binary relation on V. Then $(V, \preceq)$ is called a* partially ordered set *(poset) if $\preceq$ is a partial order (i.e. reflexive, transitive, and antisymmetric).*

Definition A.3.7 (Least upper bound and greatest lower bound)**.** *Let $(V, \preceq)$ be a poset and let $S \subseteq V$. An element $x \in V$ is called an* upper bound *of S if $s \preceq x$ holds for all $s \in S$, and x is called a* least upper bound *if in addition $x \preceq y$ holds for all upper bounds y of S. A greatest lower bound of S is defined dually.*

Remark. *If a subset S has a least upper bound (or greatest lower bound), then it is unique. Hence, we can say* the *least upper bound (or greatest lower bound) of a subset, when it exists.*

For $x, y \in V$, we write $x \vee y$ ('*x join y*') and $x \wedge y$ ('*x meet y*') for the least upper bound and the greatest lower bound of $\{x, y\}$, respectively, when they exist.

Definition A.3.8 (Lattice as poset)**.** *A poset $(V, \preceq)$ is called a* lattice *if for all $x, y \in V$, $x \vee y$ and $x \wedge y$ exist.*

Lattices as algebraic structures

Lattices can also be defined as algebraic structures. Let V be a set and $\vee : V \times V \to V$ be an arbitrary binary operator on V. Then $\vee$ is said to be:

associative if $(x \vee y) \vee z = x \vee (y \vee z)$ for all $x, y, z \in V$;

commutative if $x \vee y = y \vee x$ for all $x, y \in V$;

idempotent if $x \vee x = x$ for all $x \in V$.

Definition A.3.9 (Lattice as algebraic structure)**.** *A* lattice *is a set V equipped with two binary operators $\vee$ and $\wedge$ on V that are associative, commutative, idempotent, and satisfy the* absorption identities*: for all $x, y \in V$, $x \vee (x \wedge y) = x$ and $x \wedge (x \vee y) = x$.*

Remark. *Notice that associativity, commutativity, and idempotency concern only* one *operator, while the absorption identities are the only properties that relate the two operators of a lattice.*

Lattices as both ordered sets and algebraic strctures

The two definitions of lattice actually define the same notion.

On one hand, a lattice as algebraic structure $(V, \wedge, \vee)$ induces a partial order $\preceq$ on V by

$$\forall x, y \in V, x \preceq y \iff x \wedge y = x \iff x \vee y = y.$$

In addition, $(V, \preceq)$ is a lattice as ordered set, with $\vee$ and $\wedge$ as least upper bound and greatest lower bound operators with respect to $\preceq$.

On the other hand, a lattice $(V, \preceq)$ as ordered set is equipped with two binary operators $\vee$ (least upper bound) and $\wedge$ (greatest lower bound). These two operators are associative, commutative, idempotent, and satisfy the absorption identities. Hence, $(V, \vee, \wedge)$ is a lattice as algebraic structure.

Bounded and distributive lattices

In the main text, we denote a lattice by $(V, \preceq, \wedge, \vee)$, or simply $(V, \wedge, \vee)$ when the partial order is not important. We mostly use lattices that are bounded and distributive.

Definition A.3.10 (Bounded lattice)**.** *A lattice V is said to be* bounded *if there are elements $\bot, \top \in V$ (called* bottom *and* top*, respectively) satisfying $\bot \preceq x$ and $x \preceq \top$ for all $x \in V$.*

Definition A.3.11 (Distributive lattice)**.** *A lattice V is called* distributive *if $\vee$ and $\wedge$ distribute over each other, i.e. for all $x, y, z \in V$, it holds that*

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z),$$
$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$$

Example. *Let S be a set and let $\mathcal{P}(S)$ denote its powerset. Then $(\mathcal{P}(S), \subseteq, \cap, \cup)$ is a bounded distributive lattice with set inclusion $\subseteq$ as partial order, set intersection $\cap$ and set union $\cup$ respectively as meet and join, $\emptyset$ as $\bot$, and S as $\top$.*

In a distributive lattice, applying distributivity twice yields that for all $x, y, z \in V$,

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) = x \vee \big(y \wedge (x \vee z)\big), \tag{A.1}$$
$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) = x \wedge \big(y \vee (x \wedge z)\big). \tag{A.2}$$

which are usually referred to as *insertion identities*. These identities are very helpful in establishing results in the main text.

A.4 Rules of Bridge

We briefly present the rules of the game of Bridge, so that readers can have an intuition about its gameplay. For details, one can for example consult the Wikipedia page about contract Bridge, or the book by Fédération Française de Bridge (2013, in French only), an introduction to Bridge of excellent pedagogical quality.

A deal of Bridge begins by dealing a standard deck of 52 cards to 4 players (2 teams of 2 players); each player receives 13 cards privately. Then, the game proceeds in two phases:

Bidding During the bidding phase, players take turns to give some information about their hidden hand to their partner (but also to their opponents) in order to find the best contract to play. Informally, a contract consists of a suit chosen to be the trump suit, and a number of tricks to be taken. We do not dwell on the details of the bidding phase; it suffices to know that each team can use their own bidding conventions, but all such conventions are required to be common knowledge (i.e. for each team, their opponents know the conventions of the team, the team know that their opponents know their conventions, ad infinitum).

Card play The player who has won the bidding becomes *declarer* of the deal, and their two opponents (on the left-hand side and on the right-hand side) become *defenders*. The left-hand defender chooses a card, called the lead, to begin the first trick. After the lead is chosen and revealed to everyone on the table, the partner of declarer, called *dummy*, lays down their hand on the table to reveal it to everyone. The card play proceeds as a trick-taking game: during each trick, a card is played from each hand clockwise, and the hand that plays the winning card of the trick begins the next trick; in addition, declarer controls which card to play from dummy's hand during each trick. After 13 tricks have been played, declarer wins if they have collected enough tricks to fulfil their contract.

The inspiration of many subjects treated in this dissertation comes from the card play phase of Bridge, which can be considered as a game with incomplete information in which MAX (declarer) has single-agent incomplete information (declarer does not see defenders' hands), MIN (defenders) have multi-agent incomplete information (defenders do not see each other's hand, nor declarer's hand), there is no chance node (except for the initial dealing of cards), and there are only public actions (each card played by a player is observed by everyone in the game).

A.4.1 Bridge deals

In this subsection, we present a few deals that illustrate certain concepts that we have discussed in the main text.

Deal 1: non-locality

In Deal 1, West cashes three rounds of clubs, then plays a fourth round. For declarer, it is natural to adopt the following line: ruff high in dummy, then draw trumps in two or three rounds and claim the rest. With this line, declarer wins whenever the trumps are not 4-0, which means more than 90% of all possible cases. Hence, this line since far superior to either the spade finesse or the diamond finesse, both cover around 50% of all possible cases. However, the first line is actually a Greek gift from West. Normally,

Deal 1 A Bridge deal from *Les donnes du Monde* (N°541) showcasing the phenomenon of non-locality.

	♠ AQJ2 ♡ AJ4 ◇ KJ5 ♣ 1092					
♠ K6 ♡ 10853 ◇ Q104 ♣ AKQ4	N W E S	♠ 109754 ♡ - ◇ 98632 ♣ J83	West 2♡ Pass Pass	North Pass 4♡	East 2NT Pass	South 3◇ Pass
	♠ 83 ♡ KQ9762 ◇ A7 ♣ 765					

West would not have incentive to give you a ruff and discards. But in this particular deal, West knows that both of your finesse will succeed; they thus offer you a losing option to take the ruff and discard, thereby promoting their fourth trump to the setting trick.

Deal 2: opponent-model search

Deal 2 A Bridge deal from Bourke (2005, Chapter 10) in which declarer exploits the signals between defenders.

	♠ J9 ♡ 95 ◇ K754 ♣ 97532					
♠ A63 ♡ Q10643 ◇ 1092 ♣ K6	N W E S	♠ 10872 ♡ J7 ◇ J83 ♣ Q1084	West - Pass Pass Pass	North - 2◇ 3NT	East - Pass Pass	South 2♣ 2NT Pass
	♠ KQ54 ♡ AK82 ◇ AQ6 ♣ AJ					

In Deal 2, declarer performs an opponent-model search by exploiting the signals between defenders. After the lead of diamond to East's jack of hearts and to declarer's king of hearts, declarer only has six top tricks. The best shot to develop 3 additional tricks is to finesse against the 10 of spades or to hope for a 3-3 split for diamonds. There is no way to accumulate these two opportunities due to the lack of entries. However, it could be beneficial to lead the queen of diamond in the second trick. Both defenders will probably give an honest signal since their partner could have the ace of diamond. Hence, if both defenders follow low, declarer can assume a 3-3 split for diamonds. Otherwise, declarer plays spade to dummy's 9.

Deal 3: recursive reasoning

Deal 3 A Bridge deal from Karpin (1977, p.266) that exhibits a recursive reasoning at level-3.

	♠ Q75 ♡ 8 ◇ AQ1097 ♣ AK106					
♠ J ♡ AJ109764 ◇ 6 ♣ QJ94	N W E S	♠ 106 ♡ K52 ◇ 8432 ♣ 8752	West 4♡ Pass Pass	North Pass 6♠ Pass	East Pass Pass Pass	South 4♠ Pass Pass
	♠ AK98432 ♡ Q3 ◇ KJ5 ♣ 3					

In Deal 3,[5] by a brilliant defensive false-card, the East defender succeeded in creating a tread of thought in declarer's mind which led to declarer's defeat in a cold six-spade contract.

West opened the ace of hearts, upon which East, dropped the king! West, of course, continued with another heart, which was ruffed by dummy's queen (to prevent the "obvious" overruff.) When East followed to the second round of hearts, South was certain that the only plausible excuse for East's false-card was that East possessed the J-10-6 of spades. So the seven of spades was led from dummy, and finessed! West's singleton jack took the setting trick.

Before criticising declarer, remember one thing: East would also have made the false-card if he *had* had the J-10-6 of trumps; declarer would then have become a temporary genius instead of a gullible victim.

[5]The narrative for this deal is also taken from Karpin (1977, p.267).

Appendix B

Proofs

B.1 Proofs for Chapter 4

B.1.1 Lemmas for Proposition 4.2.4

Recall that for all $v \in V$ in a game of no chance, $\Sigma^{\mathrm{P}}_{+,v} \subseteq \Sigma^{\mathrm{P}}_{+}$ is the set of pure strategies of MAX under which v is reachable, similarly for $\Sigma^{\mathrm{P}}_{-,v} \subseteq \Sigma^{\mathrm{P}}_{-}$. Although we will not formally prove it here, in a game with multi-agent perfect recall, the sets $\Sigma^{\mathrm{P}}_{+,v}$ and $\Sigma^{\mathrm{P}}_{-,v}$ are non-empty for all vertices v. To see this, notice that the path from the root to a vertex intersects every information set of MAX or of MIN at most once, which means the path induces a well-defined pure strategy along the path for both MAX and MIN; see also the proof of Lemma B.1.3.

We first show that if a vertex is reachable under both a pure strategy of MAX and under a pure strategy of MIN, then this vertex is actually reached when both strategies are implemented.

Lemma B.1.1. *In a two-player EFG of no chance, if $s_+ \in \Sigma^{\mathrm{P}}_{+,v}$ and $s_- \in \Sigma^{\mathrm{P}}_{-,v}$ for a certain $v \in V$, then v is in the playout under the pure strategy profile (s_+, s_-).*

Proof. Let $v \in V$. We show that if $(s_+, s_-) \in \Sigma^{\mathrm{P}}_{+,v} \times \Sigma^{\mathrm{P}}_{-,v}$, then v is in the playout under this profile. Let v^* be the lowest common vertex between the path from the root r to v and the playout under the profile (s_+, s_-). v^* is well-defined since the root is reached during this playout. We will show that $v^* = v$.

Suppose by contradiction that this is not the case. Let v' be the successor of v^* in the path from r to v, which exists since $v^* \neq v$. Suppose without loss of generality that v^* is a decision node of MAX. Then under s_+, MAX does not take the unique action leading from v^* to v' at MAX's information set containing v^*. Then, no matter what pure strategy MIN implements, v' cannot be reached when MAX plays s_+. As a result, v' is not reachable under s_+, and *a fortiori* neither is v, which is a descendant of v'. This contradicts the assumption that $s_+ \in \Sigma^{\mathrm{P}}_{+,v}$.

Therefore, we must have $v^* = v$, hence v is in the playout under the profile (s_+, s_-). □

Lemma B.1.2. *In a two-player EFG of no chance and with multi-agent perfect recall, if v is a decision node of MAX, then $\Sigma^{\mathrm{P}}_{-,v} = \Sigma^{\mathrm{P}}_{-,v'}$ for all children v' of v; if v is a decision node of MIN, then $\Sigma^{\mathrm{P}}_{+,v} = \Sigma^{\mathrm{P}}_{+,v'}$ for all children v' of v.*

Proof. We only prove the first statement since the second one is completely symmetric. Let v be a decision node of MAX in a game of no chance and v' be a child of v.

- Let $s_- \in \Sigma^{\mathrm{P}}_{-,v'}$. Since v' is reachable, there is $s_+ \in \Sigma^{\mathrm{P}}_{+}$ such that v' is in the playout under (s_+, s_-). Then necessarily v, the parent of v', is also reached in the playout under this profile, hence v is reachable under s_- and $s_- \in \Sigma^{\mathrm{P}}_{-,v}$.

- Let $s_- \in \Sigma^{\mathrm{P}}_{-,v}$. Let $s_+ \in \Sigma^{\mathrm{P}}_{+}$ such that v is in the playout under (s_+, s_-). Consider the strategy s'_+ that differs from s_+ only at MAX's information set IS_+ containing v, at which s'_+ chooses the unique action leading from v to v'. Then under (s'_+, s_-), v is also reached since the path from the root to v intersects IS_+ only once due to MAX's multi-agent perfect recall. Hence, by the definition of s'_+, v' is reached under (s'_+, s_-), proving that $s_- \in \Sigma^{\mathrm{P}}_{-,v'}$.

Therefore, $\Sigma^{\mathrm{P}}_{-,v} = \Sigma^{\mathrm{P}}_{-,v'}$. □

Lemma B.1.3. *In a two-player EFG of no chance in which MAX has PR and MIN has MA-PR, if an information set IS_+ of MAX is reachable under a pure strategy s_+ of MAX, then every $v \in IS_+$ is reachable under s_+.*

Proof. Let IS_+ be an information set of MAX that is reachable under a pure strategy s_+ of MAX. Let $v \in IS_+$ be a vertex reachable under s_+, and let $v' \in IS_+$ be an arbitrary vertex in the same information set. We will show that v' is also reachable under s_+.

The conclusion trivially holds if $v = v'$. Hence, we suppose that $v \neq v'$. Let s_- be a pure strategy of MIN such that v is in the playout under (s_+, s_-). We will show how to modify s_- to get another pure strategy of MIN s'_- such that v' is in the playout under (s_+, s'_-).

First, let v^* be the lowest common ancestor[1] of v and v'. Then v^* is necessarily a decision node of MIN since otherwise there would be two distinct paths from v^* to MAX's information set IS_+ such that MAX chooses a different action at v^* in these two paths, contradicting MAX's perfect recall. Clearly, v^* is reached under (s_+, s_-) since its descendant v is reached under this profile.

Consider the unique path from v^* to v'. For each node v_- of MIN in this path, we modify the action assigned by s_- at MIN's information set containing v_- to be the unique action leading to v_-'s successor in the path from v^* to v'. Next, we prove that this modification yields a new pure strategy s'_-, and that s'_- is what we are looking for.

Well-definedness of s'_- To show that the aforementioned modifications to s_- yields a new pure strategy of MIN, it suffices to show that s_- has been modified at most once at each information set of MIN. This trivially follows from MIN's multi-agent perfect recall: the path from v^* to v' intersects each of MIN's information set at most once.

Reachability of v' under (s_+, s'_-) We now aim to prove that v' is reached under (s_+, s'_-). This follows from the two observations below:

- First, v^* is still reached under (s_+, s'_-). The reason is that the path from the root r to v^* cannot intersect any information set intersected by the path from v^* to v' (except the one containing v^*), otherwise MIN does not have multi-agent perfect recall. Hence, s'_- chooses the same action as s_- at any MIN's ancestor node of v^*, which means the playout under (s_+, s'_-) is exactly the same as the one under (s_+, s_-) at least until v^* is reached.

- Secondly, the path from v^* to v' is part of the playout under (s_+, s'_-). Suppose by contradiction that this is not the case, then the path from v^* to v' diverges from the path in the playout starting at v^*. Let v'' be the lowest vertex shared by these two paths. Then v'' cannot be a decision node of MIN, since s'_- is defined explicitly such that at any node of MIN in the path from v^* to v', MIN takes the same action under s'_- as the action they would take in this path. Hence, v'' is a decision node of MAX. By MAX's perfect recall, the path from v^* to v must intersect the information set of MAX containing v'', as the path from v^* to v' does (since v and v' belong to the same information set of MAX), and MAX chooses the same action at this information set in these two paths, which is the same as the action chosen by s_+ at v''. This means v'', which is different from v' by our assumption, cannot be the lowest common vertex between the path from

[1] In a tree, the lowest common ancestor of v and v' is the unique vertex v^* that is both a common ancestor of v and v', and the lowest one: for every common ancestor v'' of v and v', v'' is also an ancestor of v^*.

v^* to v' and the playout from v^* under (s_+, s'_-), contradicting the definition of v''. As a result, these two paths cannot diverge, and v' is indeed in the playout under (s_+, s'_-).

In conclusion, v' is reached under (s_+, s'_-). Since v' is an arbitrary vertex in IS_+, this means all vertices in IS_+ are reachable under s_+. □

Lemma B.1.4. *In a two-player EFG of no chance in which MAX has PR and MIN has MA-PR, if v and v' belong to the same information set IS_+ of MAX, then $\Sigma^{\mathrm{P}}_{+,v} = \Sigma^{\mathrm{P}}_{+,v'}$. Furthermore, for every action $a \in A_+$, we have $\Sigma^{\mathrm{P}}_{+,a(v)} = \Sigma^{\mathrm{P}}_{+,a(v')}$.*[2]

Proof. This is a direct consequence of Lemma B.1.3.

Indeed, let v and v' be two vertices in the same information set IS_+ of MAX. Let $s_+ \in \Sigma^{\mathrm{P}}_{+,v}$. Since v, and *a fortiori* IS_+, is reachable under s_+, and v' is in IS_+, we deduce by Lemma B.1.3 that v' is also reachable under s_+, which means $s_+ \in \Sigma^{\mathrm{P}}_{+,v'}$. Hence, $\Sigma^{\mathrm{P}}_{+,v} \subseteq \Sigma^{\mathrm{P}}_{+,v'}$. By inverting the roles between v and v' in the previous argument, we also get $\Sigma^{\mathrm{P}}_{+,v'} \subseteq \Sigma^{\mathrm{P}}_{+,v}$, thus establishing the first part of the statement.

For the second part, notice that $\Sigma^{\mathrm{P}}_{+,a(v)}$ contains exactly those strategies in $\Sigma^{\mathrm{P}}_{+,v}$ that choose action a at IS_+, and similarly for $\Sigma^{\mathrm{P}}_{+,a(v')}$. Therefore, $\Sigma^{\mathrm{P}}_{+,a(v)}$ is the same set as $\Sigma^{\mathrm{P}}_{+,a(v')}$ since $\Sigma^{\mathrm{P}}_{+,v} = \Sigma^{\mathrm{P}}_{+,v'}$. □

Armed with these lemmas regarding the properties of reachability in a game of no chance, we can prove the lemmas used in the proof of Proposition 4.2.4 (page 36).

Lemma 4.2.6. *For every vertex $v \in V$, it holds that*

$$ev(v) \geq \max_{s_+ \in \Sigma^{\mathrm{P}}_{+,v}} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-). \tag{4.3}$$

Proof. We prove this lemma by structural induction on the game tree. Let $v \in V$ be an arbitrary vertex in the tree.

If v is a leaf, then $ev(v) = u_+(v)$. Let $s_+ \in \Sigma^{\mathrm{P}}_{+,v}$. Since v is reachable under s_+, there is $s^*_- \in \Sigma^{\mathrm{P}}_-$ such that the leaf v is reached in the playout under (s_+, s^*_-), which means this profile yields a payoff of $u_+(v)$ for MAX. Hence, $\min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-) \leq u_+(v)$, from which we deduce that (4.3) holds.

If v is a decision node of MIN, then $\Sigma^{\mathrm{P}}_{+,v'} = \Sigma^{\mathrm{P}}_{+,v}$ for all $v' \in \mathrm{C}(v)$ by Lemma B.1.2. By the induction hypothesis, we have

$$ev(v') \geq \max_{s_+ \in \Sigma^{\mathrm{P}}_{+,v'}} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-) = \max_{s_+ \in \Sigma^{\mathrm{P}}_{+,v}} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-),$$

which, combined with $ev(v) = \min_{v' \in \mathrm{C}(v)} ev(v')$, yields (4.3).

Now consider the case in which v is a decision node of MAX. Let IS_+ be MAX's information set that contains v. Then we have

$$\begin{aligned}
ev(v) &= \max_{a \in A_+} \min_{v' \in IS_+} ev(a(v')) \\
&\geq \max_{a \in A_+} \min_{v' \in IS_+} \max_{s_+ \in \Sigma^{\mathrm{P}}_{+,a(v')}} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-) \\
&= \max_{a \in A_+} \max_{s_+ \in \Sigma^{\mathrm{P}}_{+,a(v)}} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-) \\
&= \max_{s_+ \in \Sigma^{\mathrm{P}}_{+,v}} \min_{s_- \in \Sigma^{\mathrm{P}}_-} \mathcal{U}_+(s_+, s_-),
\end{aligned}$$

[2] $a(v)$ and $a(v')$ denote the vertex reached by taking action a at v and at v', respectively.

where the first line is by the definition of $ev(v)$, the second line is by the induction hypothesis applied to $ev(a(v'))$, the third line is by Lemma B.1.4 ($\Sigma^{\mathrm{P}}_{+,a(v')}$ is the same set when v' ranges over IS_+), the fourth line is by the fact that

$$\Sigma^{\mathrm{P}}_{+,v} = \bigcup_{a \in A_+} \Sigma^{\mathrm{P}}_{+,a(v)},$$

which holds in any game of no chance, even without multi-agent perfect recall. Therefore, (4.3) holds for v. □

Lemma 4.2.7. *Let $s^*_+ \in \Sigma^{\mathrm{P}}_{+,ev}$. For every vertex v reachable under s^*_+, it holds that*

$$ev(v) \leq \min_{s_- \in \Sigma^{\mathrm{P}}_{-,v}} \mathcal{U}_+(s^*_+, s_-). \tag{4.5}$$

Proof. Let $s^*_+ \in \Sigma^{\mathrm{P}}_{+,ev}$. We prove this lemma by structural induction on the game tree. Let $v \in V$ be an arbitrary vertex that is reachable under s^*_+.

If v is a leaf, then $ev(v) = u_+(v)$. Let $s_- \in \Sigma^{\mathrm{P}}_{-,v}$. Since the leaf v is reachable both under s^*_+ and under s_-, by Lemma B.1.1 v is reached in the playout under (s^*_+, s_-). Hence, the leaf v is the outcome under (s^*_+, s_-), which means $\mathcal{U}_+(s^*_+, s_-) = u_+(v)$. Hence, $\min_{s_- \in \Sigma^{\mathrm{P}}_{-,v}} \mathcal{U}_+(s^*_+, s_-) = u_+(v)$ and (4.5) holds.

If v is a decision node of MIN, then every child $v' \in \mathrm{C}(v)$ is also reachable under s^*_+ (which is shown in the proof of Lemma B.1.2). Hence, the induction applies to all children of v, and we have

$$ev(v) = \min_{v' \in \mathrm{C}(v)} ev(v') \leq \min_{v' \in \mathrm{C}(v)} \min_{s_- \in \Sigma^{\mathrm{P}}_{-,v'}} \mathcal{U}_+(s^*_+, s_-) = \min_{s_- \in \Sigma^{\mathrm{P}}_{-,v}} \mathcal{U}_+(s^*_+, s_-),$$

where the first equality is by the definition of $ev(v)$, the inequality is by the induction hypothesis applied to $ev(v')$, and the second equality is by the fact that

$$\Sigma^{\mathrm{P}}_{-,v} = \bigcup_{v' \in \mathrm{C}(v)} \Sigma^{\mathrm{P}}_{-,v'},$$

which holds in any game of no chance, even without multi-agent perfect recall. Therefore, (4.5) also holds for v.

Now consider the case in which v is a decision node of MAX. Let IS_+ be MAX's information set that contains v. We denote by $a^* \in A_+$ the action chosen by s^*_+ at IS_+. By the definition of s^*_+, a^* is an action such that the maximum in (4.1) is achieved. Hence, by the definition of $ev(v)$,we have

$$ev(v) = \max_{a \in A_+} \min_{v' \in IS_+} ev(a(v')) = \min_{v' \in IS_+} ev(a^*(v')) \leq ev(a^*(v)).$$

Since v is reachable under s^*_+, which takes action a^* at IS_+, $a^*(v)$ is also reachable under s^*_+. As a consequence, the induction applies to $a^*(v)$, and we have

$$ev(v) \leq ev(a^*(v)) \leq \min_{s_- \in \Sigma^{\mathrm{P}}_{-,a^*(v)}} \mathcal{U}_+(s^*_+, s_-) = \min_{s_- \in \Sigma^{\mathrm{P}}_{-,v}} \mathcal{U}_+(s^*_+, s_-),$$

where the last equality is by Lemma B.1.2 ($\Sigma^{\mathrm{P}}_{-,a^*(v)} = \Sigma^{\mathrm{P}}_{-,v}$ since v is a decision node of MAX). Therefore, (4.5) holds for v. □

B.1.2 Compiling away non-Boolean payoffs

Some proofs in this dissertation involve EFGs of chance with non-Boolean but integer payoffs. We will show that these payoffs can always be compiled into Boolean ones with the help of chance nodes.

Lemma B.1.5. *Let G be a two-player EFG of chance with only integer payoffs, and let m be a rational number. Then there is a Boolean game G' and a rational number m', constructible in time polynomial in the size of G and m, such that*

- *Every player has the same degree of imperfect information and the same set of strategies in G' as in G.*
- *With respect to every possible set of strategies Σ_+ and Σ_-,[3] the maxmin value for MAX in G' is m' if and only if the maxmin value for MAX in G is m.*

Proof. Recall that the maxmin value of a game with respect to Σ_+ and Σ_- is defined as

$$\underline{v}_+(\Sigma_+, \Sigma_-) \coloneqq \max_{\varsigma_+ \in \Sigma_+} \min_{\varsigma_- \in \Sigma_-} \mathcal{U}_+(\varsigma_+, \varsigma_-).$$

Transforming the payoffs into rational numbers in the interval $[0, 1]$ Every affine transformation with positive scaling for all payoffs induces the same affine transformation on the maxmin value: for all $a > 0$ and $b \in \mathbb{R}$,

$$\max_{\varsigma_+ \in \Sigma_+} \min_{\varsigma_- \in \Sigma_-} \big(a \cdot \mathcal{U}_+(\varsigma_+, \varsigma_-) + b\big) = a \cdot \underline{v}_+(\Sigma_+, \Sigma_-) + b$$

holds for every possible set of strategies Σ_+ and Σ_-. We can exploit this property to transform all integer payoffs of G into rational numbers in the interval $[0, 1]$ with binary expansion of polynomial length in the size of G.

Let $n \geq 0$ be the largest payoff in G in terms of absolute value, which means all payoffs in G are integers in $[-n, n]$. Let $d \in \mathbb{N}$ be the smallest integer such that $2^d \geq 2n$. Notice that the value of d is linearly bounded by the size of G. By adding n to all payoffs in G, we shift all payoffs into integers in $[0, 2n]$; then by dividing all payoffs by 2^d, we transform the payoffs into rational numbers in the interval $[0, 1]$ with a binary expansion over at most d digits. If the initial game has a maxmin value of m, then the new game G' has a maxmin value of $m' = (m + n)/2^d$.

Further transforming the payoffs into Boolean values Now, we show how to get rid of the fractional payoffs in G' with the help of chance nodes. Let $x = i/2^d$ with $0 \leq i \leq 2^d$. Then a leaf with a payoff of x can be replaced by a chance tree T_x of depth d that contains only chance nodes with uniform Bernoulli distribution and Boolean payoffs in the following way: if the d-digit binary representation of i reads $i_1 i_2 \cdots i_d$, then the right node of the chance tree at depth $1 \leq j \leq d$ has value i_j, and the left node at depth d has value 0. See Figure B.1 for an example with $x = 5/8$ (i.e. $d = 3$ and $i = 5$ with binary representation 101); all internal nodes are binary chance nodes that lead to either child with probability $1/2$.

By construction, T_x for $x = i/2^d$ is of size $O(d)$, and the root of T_x has an expected value of x. Since there is neither decision node of MAX nor of MIN in T_x, one can replace a leaf with a payoff of x in G' by T_x without changing the expected payoff $\mathcal{U}$

[3] By every possible set of strategies of MAX, we mean every subset of $\Sigma_+^{\mathrm{P}} \cup \Sigma_+^{\mathrm{M}} \cup \Sigma_+^{\mathrm{B}}$ (i.e. a set containing arbitrary pure, mixed, and behaviour strategies); likewise for MIN.

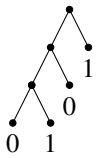


Figure B.1: A Boolean chance tree that represents the value $5/8$.

under any strategy profile of the players, and *a fortiori* without changing the maxmin value of G'.

To summarise, after obtaining G' from G by an affine transformation on the payoffs, we replace each leaf with a payoff of the form $x = i/2^d$, where $0 \leq i \leq 2^d$, by a Boolean chance tree T_x to obtain the game G''. Then, the whole construction is polynomial in G and m, and G'' and m' satisfy all the requirements. □

Remark. *By an affine transformation, any game with only two possible values as payoffs can be transformed into a Boolean game.*

B.1.3 Tseitin transformation for compact Boolean games

Lemma 4.3.3. *Let $\gamma = \langle X, \mathrm{P}, \mathrm{D}, \varphi_+ \rangle$ be a CBG. Then there exists a CBG $\gamma' := \langle X', \mathrm{P}', \mathrm{D}', \varphi'_+ \rangle$ with φ'_+ in CNF (respectively DNF, ROBDD), which has the same maxmin value as γ, and such that (i) γ' is of no chance if and only if γ is of no chance; (ii) MAX and MIN have the same degree of imperfect information in γ' as in γ. Furthermore, γ' can be constructed from γ in polynomial time.*

Proof. Let $\exists y_1 \cdots \exists y_p \, \psi_+(X \cup \{y_1, \ldots, y_p\})$ be a formula which is satisfied by an assignment to X if and only if φ_+ outputs 1 on this assignment, and in which ψ_+ is in CNF, $y_1, \ldots, y_p$ are variables not in X and p is polynomial in the size of φ_+. Such a formula can be obtained from φ_+ in polynomial time using the Tseitin transformation (Tseitin, 1983).

To build a CBG γ' with a CNF goal from γ, we define $X' := X \cup \{y_1, \ldots, y_n\}$ (with the ordering from X then $y_1, \ldots, y_n$), $\mathrm{P}'(x) := \mathrm{P}(x)$ for $x \in X$ and $\mathrm{P}'(y_i) := +$ for $i = 1, \ldots, p$, $\mathrm{D}'(x) := \mathrm{D}(x)$ for $x \in X$ and $\mathrm{D}'(y_i) := X \cup \{y_1, \ldots, y_{i-1}\}$ for $i = 1, \ldots, p$, and finally $\varphi'_+ := \psi_+$. Observe that the construction is polynomial time, does not introduce any chance variable, and preserves PI, PR, and MA-PR for both players (since all information is revealed at the extra nodes). We now show that γ' has the same maxmin value as γ.

Given a strategy ς_+ for MAX in γ, we simply define ς'_+ for MAX in γ' to extend ς_+ by choosing $y_1, \ldots, y_p$ such that ψ_+ is true at all leaves of γ at which φ_+ outputs 1, and arbitrary values at the other leaves. This can be done because in γ', all information is revealed to MAX at the leaves of γ. Moreover, ς'_+ is pure if and only if so is ς_+. Finally, by construction, this ensures that the utility of ς'_+ in γ' is the same as the utility of ς_+ in γ, against any strategy of MIN (observe that the set of strategies for MIN is the same in γ and γ'), and hence that the maxmin value is preserved.

For DNF, we use the dual construction. Given a CBG with an arbitrary goal φ_+, we first build a formula $\exists y_1 \cdots \exists y_p \psi_+(X \cup \{y_1, \ldots, y_p\})$ as above, where ψ_+ is in CNF, but which is satisfied by an assignment to X if and only if φ_+ outputs 0 on this assignment; such a formula can be obtained from $\neg\varphi_+$ using the Tseitin transformation.

Then we build a DNF χ_+ equivalent to $\neg\psi_+$ using De Morgan's laws. Finally, we define γ' by $X' := X \cup \{y_1, \ldots, y_p\}$, $P'(x) := P(x)$ for $x \in X$ and $P'(y_i) := -$ for $i = 1, \ldots, p$, D' as in the construction for CNF, and finally $\varphi'_+ := \chi_+$. Then MAX has the same set of strategies in γ' as in γ. Now given a strategy ς_- for MIN in γ, we define ς'_- for MIN in γ' to extend ς_- by choosing $y_1, \ldots, y_p$ such that χ_+ is false at all leaves of γ at which φ_+ outputs 0, and arbitrary values at other leaves; indeed, if and only if φ_+ outputs 0, then by construction there exists $y_1, \ldots, y_p$ such that ψ_+ is true, that is, such that χ_+ is false. It follows that MIN can force a loss for MAX in γ' exactly at those leaves in which MAX loses in γ, as desired.

Finally, for ROBDD, we use the construction given by Darwiche and Marquis (2002, middle of page 258): for a given DNF φ_+, the construction introduces auxiliary variables $y_1, \ldots, y_p$ not in X, and builds in polynomial time an ROBDD ψ_+ over $X \cup \{y_1, \ldots, y_p\}$ such that $\exists y_1 \cdots \exists y_p \, \psi_+(X \cup \{y_1, \ldots, y_p\})$ is equivalent to φ_+. Hence, with the same reasoning as for CNF above, we conclude that a CBG with a DNF goal can be reformulated in polynomial time into a CBG with an ROBDD goal. By the previous paragraph, it follows that any CBG can be reformulated into a CBG with an ROBDD goal. □

B.2 Proofs for Chapter 6

Proposition 6.4.2. *If h is an admissible heuristic function for G and V, then for all nodes n of G and all $\alpha, \beta \in V$, we have*

$$\beta \wedge \big(\alpha \vee f(n, \alpha, \beta)\big) = \beta \wedge \big(\alpha \vee \mathsf{val}(n)\big). \tag{6.9}$$

Proof. The proof is based on structural induction on n. We will only focus on OR-nodes since the case for AND-nodes is completely symmetric. So let n be an OR-node and let us consider the execution of the function call `AlphaBeta`(n, α, β).

If n is a leaf, then $f(n, \alpha, \beta) = h(n, \alpha, \beta) = \mathsf{val}(n)$ since h is admissible, hence (6.9) holds trivially.

Now consider an internal OR-node n. Let $b \geq 1$ be the number of children of n and let $n_1, \ldots, n_b$ be the children of n, listed in the same order as in Algorithm 4. By definition, $\mathsf{val}(n) = \bigvee_{j=1}^{b} \mathsf{val}(n_j)$. We assume by induction that all function calls on the children of n satisfy (6.9).

Let k be the index of the loop during which a break happens (i.e. a cut is found). If no break happens, then k is taken to be $b + 1$. For $0 \leq i < k$, let v_i and α_i denote the value of v and α after the i-th loop[4]. Then we have

$$v_i = I \vee \bigvee_{j=1}^{i} f(n_j, \alpha_j, \beta),$$

where $I = h(n, \alpha, \beta)$. In particular, $v_0 = I$. In addition, $\alpha_0 = \alpha$, and $\alpha_i = \alpha \vee v_{i-1}$ for $1 < i < k$.

To prove (6.9) for node n, we need the following lemma:

Lemma B.2.1. *For every $0 \leq i < k$, we have*

$$\beta \wedge (\alpha \vee v_i) = \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{i} \mathsf{val}(n_j)\right). \tag{B.1}$$

Proof. We prove (B.1) by an induction on i. When $i = 0$, both sides of (B.1) equal $\beta \wedge (\alpha \vee I)$, hence the equality holds.

For $i \geq 1$, $f(n_i, \alpha_i, \beta)$ satisfies (6.9) by induction from the main proposition, which means

$$\beta \wedge \big(\alpha_i \vee f(n_i, \alpha_i, \beta)\big) = \beta \wedge \big(\alpha_i \vee \mathsf{val}(n_i)\big). \tag{B.2}$$

Since $\alpha_i = \alpha \vee v_{i-1}$, we have

$$\begin{aligned}
\beta \wedge (\alpha \vee v_i) &= \beta \wedge \big(\alpha \vee v_{i-1} \vee f(n_i, \alpha_i, \beta)\big) \\
&= \beta \wedge \big(\alpha \vee v_{i-1} \vee \mathsf{val}(n_i)\big) \\
&= \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{i-1} \mathsf{val}(n_j) \vee \mathsf{val}(n_i)\right) \\
&= \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{i} \mathsf{val}(n_j)\right),
\end{aligned}$$

where the second line is due to (B.2), the third line is due to distributivity and (B.1) applied to v_{i-1}. Therefore, (B.1) holds for all $0 \leq i < k$. □

[4] By after the 0th loop, we mean before the beginning of the first loop.

Now we can complete the proof of Proposition 6.4.2. For a non-leaf OR-node n with $b \geq 1$ children, two cases are possible:

- No break has taken place, which means $k = b + 1$ and the algorithm has looped through all children of n. Then we have $f(n, \alpha, \beta) = v_b$. Plugging $i = b$ into (B.1), we get

$$\begin{aligned}\beta \wedge (\alpha \vee v_b) &= \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{b} \mathsf{val}(n_j)\right) \\ &= \beta \wedge \big(\alpha \vee I \vee \mathsf{val}(n)\big) \\ &= \beta \wedge \big(\alpha \vee \mathsf{val}(n)\big)\end{aligned}$$

  where the last equality is due to $I = h(n, \alpha, \beta) \preceq \mathsf{val}(n) \vee \alpha$ for an OR-node since h is admissible. Hence, (6.9) holds for node n.

- A break happens during the k-th loop, where $1 \leq k \leq b$, which means $\alpha_k = \alpha \vee v_{k-1} \succeq \beta$ (Line 10 in Algorithm 4) and $f(n, \alpha, \beta) = v_{k-1}$. On the right-hand side of (6.9), we have

$$\begin{aligned}\beta \wedge \big(\alpha \vee \mathsf{val}(n)\big) &= \beta \wedge \big(\alpha \vee I \vee \mathsf{val}(n)\big) \\ &= \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{b} \mathsf{val}(n_j)\right) \\ &\succeq \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{k-1} \mathsf{val}(n_j)\right) \\ &= \beta \wedge (\alpha \vee v_{k-1}),\end{aligned}$$

  where the first line is due to $I \preceq \alpha \vee \mathsf{val}(n)$, the second one is by the definition of $\mathsf{val}(n)$, and the last line is due to (B.1) applied to $i = k - 1$. Hence,

$$\beta \succeq \beta \wedge \big(\alpha \vee \mathsf{val}(n)\big) \succeq \beta \wedge (\alpha \vee v_{k-1}) = \beta,$$

  which means all inequalities are equalities. Hence,

$$\beta \wedge \big(\alpha \vee f(n, \alpha, \beta)\big) = \beta \wedge \big(\alpha \vee v_{k-1}\big) = \beta \wedge \big(\alpha \vee \mathsf{val}(n)\big).$$

  which means, (6.9) holds for node n.

□

Proposition 6.4.6. *If $(\underline{h}, \overline{h})$ is admissible and* Cache *is initially coherent for G and V, then for all nodes n in G and all $\alpha, \beta \in V$, we have*

$$\underline{f}(n, \alpha, \beta) \preceq \mathsf{val}(n) \preceq \overline{f}(n, \alpha, \beta). \tag{6.10}$$

Moreover, if there is an entry $(\underline{c}, \overline{c})$ in the cache for n before the call, then $\underline{c} \preceq \underline{f}(n, \alpha, \beta)$ and $\overline{f}(n, \alpha, \beta) \preceq \overline{c}$.

Proof. $\underline{c} \preceq \underline{f}(n,\alpha,\beta)$ and $\overline{f}(n,\alpha,\beta) \preceq \overline{c}$ are direct consequences of Line 31 and Line 32.

The proof of (6.10) is based on structural induction on n. We will only focus on OR-nodes, since the case for AND-nodes is completely symmetric. So let n be an OR-node and let us consider the execution of the function call $\texttt{AlphaBetaDuo}(n,\alpha,\beta)$.

If n is a leaf node, then whether or not there is an entry for n in the cache, on Line 6 we have $\underline{c} = \overline{c} = \mathsf{val}(n)$ since $(\underline{h},\overline{h})$ is admissible and Cache is coherent. Hence, the function returns immediately, so (6.10) holds and the cache remains coherent.

Otherwise, n is an internal OR-node. Again, whether or not there is an entry for n in the cache, on Line 6 and forward we have $\underline{c} \preceq \mathsf{val}(n) \preceq \overline{c}$ since $(\underline{h},\overline{h})$ is admissible and Cache is coherent.

Let $b \geq 1$ be the number of children of n. We assume by induction that all function calls on the children of n satisfy (6.10) and maintain the coherence of the cache. Let k be the index of the loop during which a break happens (if no break happens, then k is taken to be $b+1$). For $1 \leq i \leq k-1$, let $\underline{v}^i$ and $\overline{v}^i$ denote the values returned by the function call on the child n_i during the i-th loop. Then, by the induction assumption, (6.10) yields $\underline{v}^i \preceq \mathsf{val}(n_i) \preceq \overline{v}^i$ for $1 \leq i \leq k-1$.

Hence, after the loop (i.e. just before Line 31),

$$\underline{v} = \bigvee_{i=1}^{k-1} \underline{v}^i \preceq \bigvee_{i=1}^{k-1} \mathsf{val}(n_i) \preceq \bigvee_{i=1}^{b} \mathsf{val}(n_i) = \mathsf{val}(n).$$

As for $\overline{v}$, we have two cases.

- No break happens, i.e. $k = b+1$. Then

$$\overline{v} = \bigvee_{i=1}^{b} \overline{v}^i \succeq \bigvee_{i=1}^{b} \mathsf{val}(n_i) = \mathsf{val}(n).$$

- Otherwise, a break happens, and $\overline{v} = \overline{c} \succeq \mathsf{val}(n)$.

So in both cases, $\overline{v} \succeq \mathsf{val}(n)$.

Therefore, after the final updates on Line 31 and Line 32, we have $\underline{v} \preceq \mathsf{val}(n) \preceq \overline{v}$. As a result, the returned values of the function call $\texttt{AlphaBetaDuo}(n,\alpha,\beta)$ satisfy (6.10), and the cache remains coherent after the function call. □

Proposition 6.4.7. *If $(\underline{h},\overline{h})$ is admissible and* Cache *is initially coherent for G and V, then for all nodes n in G and all $\alpha,\beta \in V$ we have*

$$\beta \wedge \big(\alpha \vee \underline{f}(n,\alpha,\beta)\big) = \beta \wedge \big(\alpha \vee \overline{f}(n,\alpha,\beta)\big). \tag{6.11}$$

Proof. Again, the proof is based on structural induction on n, and we will only focus on OR-nodes since the case for AND-nodes is symmetric.

If n is a leaf node, then $\underline{f}(n,\alpha,\beta) = \overline{f}(n,\alpha,\beta) = \mathsf{val}(n)$ since $(\underline{h},\overline{h})$ is admissible and Cache is coherent. Hence, (6.11) holds.

Otherwise, let $b \geq 1$ be the number of children of n. We assume by induction that all function calls on the children of n satisfy (6.11). Let k be the index of the loop during which a break happens (if no break happens, then k is taken to be $b+1$). For $1 \leq j < k$, let $\underline{v}^j$ and $\overline{v}^j$ denote the values returned by the function call on n_j during the j-th loop. For $0 \leq i < k$, let $\underline{v}_i$, $\overline{v}_i$, and α_i denote the value of $\underline{v}$, $\overline{v}$, and α after the i-th loop. Then for $0 \leq i < k$, we have $\underline{v}_i = \bigvee_{j=1}^{i} \underline{v}^j$, $\overline{v}_i = \bigvee_{j=1}^{i} \overline{v}^j$, and

$\alpha_i = \alpha \vee \underline{c} \vee \underline{v}_i$. In particular, $\underline{v}_0 = \overline{v}_0 = \bot$ and $\alpha_0 = \alpha \vee \underline{c}$. For $1 \leq j < k$, since the function call on the child n_j has the form $\texttt{AlphaBetaDuo}(n_j, \alpha_j, \beta \wedge \overline{c})$, by (6.11), we have $\beta \wedge \overline{c} \wedge \left(\alpha_j \vee \underline{v}^j\right) = \beta \wedge \overline{c} \wedge \left(\alpha_j \vee \overline{v}^j\right)$. Hence, by distributivity, we have

$$\beta \wedge \overline{c} \wedge \bigvee_{j=1}^{k-1} \left(\alpha_j \vee \underline{v}^j\right) = \beta \wedge \overline{c} \wedge \bigvee_{j=1}^{k-1} \left(\alpha_j \vee \overline{v}^j\right). \tag{B.3}$$

In the following, we distinguish two cases: no break happens (i.e. $k = b + 1$), or a break happens (i.e. $1 \leq k \leq b$).

No break Then $k = b + 1$ and we have

$$\underline{f}(n, \alpha, \beta) = \underline{c} \vee \underline{v}_b = \underline{c} \vee \bigvee_{j=1}^{b} \underline{v}^j,$$

$$\overline{f}(n, \alpha, \beta) = \overline{c} \wedge \overline{v}_b = \overline{c} \wedge \bigvee_{j=1}^{b} \overline{v}^j.$$

First, recall the insertion identities of a distributive lattice: for all $x, y, z \in V$,

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) = x \vee \big(y \wedge (x \vee z)\big), \tag{A.1}$$

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) = x \wedge \big(y \vee (x \wedge z)\big). \tag{A.2}$$

Applying (A.1) and (A.2), one has

$$\begin{aligned}
\beta \wedge \left(\alpha \vee \overline{f}(n, \alpha, \beta)\right) &= \beta \wedge \left(\alpha \vee \left(\overline{c} \wedge \bigvee_{j=1}^{b} \overline{v}^j\right)\right) \\
&= \beta \wedge \left(\alpha \vee \left(\overline{c} \wedge \left(\alpha \vee \bigvee_{j=1}^{b} \overline{v}^j\right)\right)\right) \\
&= \beta \wedge \left(\alpha \vee \left(\beta \wedge \overline{c} \wedge \left(\alpha \vee \bigvee_{j=1}^{b} \overline{v}^j\right)\right)\right).
\end{aligned}$$

We will first focus on the term $\beta \wedge \overline{c} \wedge (\alpha \vee \bigvee_{j=1}^{b} \overline{v}^j)$. Our goal is to massage it into a form to which the induction assumption (B.3) can apply.

Recall that for $j \leq b$, $\alpha_j = \alpha_0 \vee \underline{v}_j = \alpha_0 \vee \bigvee_{l=1}^{j} \underline{v}^l$. Hence,

$$\begin{aligned}
\bigvee_{j=1}^{b} (\alpha_j \vee \overline{v}^j) &= \bigvee_{j=1}^{b} \left(\alpha_0 \vee \bigvee_{l=1}^{j} \underline{v}^l \vee \overline{v}^j\right) \\
&= \alpha_0 \vee \bigvee_{j=1}^{b} \bigvee_{l=1}^{j} \underline{v}^l \vee \bigvee_{j=1}^{b} \overline{v}^j \\
&= \alpha_0 \vee \bigvee_{j=1}^{b} \underline{v}^j \vee \bigvee_{j=1}^{b} \overline{v}^j \\
&= \alpha_0 \vee \bigvee_{j=1}^{b} \overline{v}^j,
\end{aligned}$$

where in the last equality we use the fact that by Proposition 6.4.6, we have $\underline{v}^j \preceq \overline{v}^j$ for all $j \leq b$. Similarly, we have

$$\bigvee_{j=1}^{b}(\alpha_j \vee \underline{v}^j) = \bigvee_{j=1}^{b}\left(\alpha_0 \vee \bigvee_{l=1}^{j}\underline{v}^l \vee \underline{v}^j\right) = \alpha_0 \vee \bigvee_{j=1}^{b}\underline{v}^j.$$

Hence, by $\alpha_0 = \alpha \vee \underline{c} \succeq \alpha$, the two previous equalities, distributivity, and the induction assumption (B.3),

$$\begin{aligned}
\beta \wedge \overline{c} \wedge \left(\alpha \vee \bigvee_{j=1}^{b}\overline{v}^j\right) &\preceq \beta \wedge \overline{c} \wedge \left(\alpha_0 \vee \bigvee_{j=1}^{b}\overline{v}^j\right) \\
&= \beta \wedge \overline{c} \wedge \bigvee_{j=1}^{b}(\alpha_j \vee \overline{v}^j) \\
&= \beta \wedge \overline{c} \wedge \bigvee_{j=1}^{b}(\alpha_j \vee \underline{v}^j) \\
&= \beta \wedge \overline{c} \wedge \left(\alpha_0 \vee \bigvee_{j=1}^{b}\underline{v}^j\right) \\
&\preceq \beta \wedge \left(\alpha_0 \vee \bigvee_{j=1}^{b}\underline{v}^j\right).
\end{aligned}$$

Therefore, applying again (A.1) and (A.2) yields

$$\begin{aligned}
&\beta \wedge \left(\alpha \vee \overline{f}(n,\alpha,\beta)\right) \\
&= \beta \wedge \left(\alpha \vee \left(\beta \wedge \overline{c} \wedge \left(\alpha \vee \bigvee_{j=1}^{b}\overline{v}^j\right)\right)\right) \\
&\preceq \beta \wedge \left(\alpha \vee \left(\beta \wedge \left(\alpha_0 \vee \bigvee_{j=1}^{b}\underline{v}^j\right)\right)\right) \\
&= \beta \wedge \left(\alpha \vee \alpha_0 \vee \bigvee_{j=1}^{b}\underline{v}^j\right) \\
&= \beta \wedge \left(\alpha \vee \underline{c} \vee \bigvee_{j=1}^{b}\underline{v}^j\right) \\
&= \beta \wedge \left(\alpha \vee \underline{f}(n,\alpha,\beta)\right).
\end{aligned}$$

Break Then $1 \leq k \leq b$, and we have $\overline{f}(n,\alpha,\beta) = \overline{c}$ and $\underline{f}(n,\alpha,\beta) = \underline{c} \vee \underline{v}_{k-1}$. Since a break happens during the k-th loop, according to Line 16 in Algorithm 5 we have $\beta \wedge \overline{c} \preceq \alpha_{k-1} = \alpha \vee \underline{c} \vee \underline{v}_{k-1}$. Hence, by distributivity,

$$\begin{aligned}
\beta \wedge \left(\alpha \vee \overline{f}(n,\alpha,\beta)\right) &= \beta \wedge (\alpha \vee \overline{c}) \\
&= \beta \wedge \left(\alpha \vee (\beta \wedge \overline{c})\right) \\
&\preceq \beta \wedge \left(\alpha \vee (\alpha \vee \underline{c} \vee \underline{v}_{k-1})\right) \\
&= \beta \wedge \left(\alpha \vee \underline{f}(n,\alpha,\beta)\right).
\end{aligned}$$

Conclusion No matter a break happens or not, we have

$$\beta \wedge \big(\alpha \vee \overline{f}(n, \alpha, \beta)\big) \preceq \beta \wedge \big(\alpha \vee \underline{f}(n, \alpha, \beta)\big).$$

On the other hand, by Proposition 6.4.6, $\overline{f}(n, \alpha, \beta) \succeq \underline{f}(n, \alpha, \beta)$, which means

$$\beta \wedge \big(\alpha \vee \overline{f}(n, \alpha, \beta)\big) \succeq \beta \wedge \big(\alpha \vee \underline{f}(n, \alpha, \beta)\big).$$

As a consequence, (6.11) holds. □

In the following, we prove Proposition 6.5.2. Although Proposition 6.5.2 generalises Proposition 6.4.2, we have still shown a proof of Proposition 6.4.2 before for two reasons.

- The proof of Proposition 6.4.2 is much easier to follow due to the lack of gamma; it shows the core ideas behind the inductions used for the proof, which are reused in the proof of Proposition 6.5.2;
- The presence of gamma breaks the symmetry between AND-nodes and OR-nodes since the mask γ' is combined to the initial value I by a meet $\wedge$. Hence, we need to establish the induction for both OR-nodes and AND-nodes in the proof of Proposition 6.5.2, unlike the case for other proofs in this chapter.

Proposition 6.5.2. *If h is an admissible heuristic function and g is an admissible gamma update function for G and V, then for all nodes n of G and all $\alpha, \beta, \gamma \in V$, we have*

$$\beta \wedge \big(\alpha \vee f(n, \alpha, \beta, \gamma)\big) = \beta \wedge \Big(\alpha \vee \big(g(n, \gamma) \wedge \mathsf{val}(n)\big)\Big), \tag{6.12}$$

$$\alpha \vee \big(\beta \wedge f(n, \alpha, \beta, \gamma)\big) = \alpha \vee \Big(\beta \wedge \big(g(n, \gamma) \wedge \mathsf{val}(n)\big)\Big). \tag{6.13}$$

Proof. First, we show that in a distributive lattice, (6.12) and (6.13) are equivalent. Indeed, recall the insertion identities of a distributive lattice (cf. Appendix A.3.2): for all $x, y, z \in V$,

$$x \vee (y \wedge z) = x \vee \big(y \wedge (x \vee z)\big), \tag{A.1}$$

$$x \wedge (y \vee z) = x \wedge \big(y \vee (x \wedge z)\big). \tag{A.2}$$

If (6.12) holds, then for all $\alpha, \beta, \gamma \in V$ and all nodes n of G,

$$\begin{aligned} \alpha \vee \big(\beta \wedge f(n, \alpha, \beta, \gamma)\big) &= \alpha \vee \Big(\beta \wedge \big(\alpha \vee f(n, \alpha, \beta, \gamma)\big)\Big) && \text{(B.4)} \\ &= \alpha \vee \Big(\beta \wedge \Big(\alpha \vee \big(g(n, \gamma) \wedge \mathsf{val}(n)\big)\Big)\Big) && \text{(B.5)} \\ &= \alpha \vee \Big(\beta \wedge \big(g(n, \gamma) \wedge \mathsf{val}(n)\big)\Big), && \text{(B.6)} \end{aligned}$$

where we use the insertion identity (A.1) on the first and third line, and (6.12) on the second line. Therefore, (6.12) implies (6.13). Symmetrically, one can show that (6.13) also implies (6.12), so these two equalities are equivalent.

As a consequence, we only have to prove one of (6.12) and (6.13) holds. The proof is based on structural induction. Let n be a node and $\gamma' = g(n, \gamma)$. Let us consider the execution of the function call `AlphaBetaGamma`$(n, \alpha, \beta, \gamma)$.

Base case Consider the case in which n is a leaf node. Then

$$f(n, \alpha, \beta, \gamma) = \gamma' \wedge h(n, \alpha, \beta) = g(n, \gamma) \wedge \mathsf{val}(n)$$

since h is admissible. Hence, (6.12) and (6.13) hold trivially.

Now let n be an arbitrary internal node. We assume by induction that all function calls on the children of n satisfy (6.12) (or equivalently (6.13)).

Induction for OR-nodes Consider the case in which n is an internal OR-node. Let $b \geq 1$ be the number of children of n and let $n_1, \ldots, n_b$ be the children of n, listed in the same order as in Algorithm 6. By definition, $\mathsf{val}(n) = \bigvee_{j=1}^{b} \mathsf{val}(n_j)$. Let k be the index of the loop during which a break happens (i.e. a cut is found). If no break happens, then k is taken to be $b + 1$. For $0 \leq i < k$, let v_i and α_i denote the value of v and α after the i-th loop[5]. Then we have

$$v_i = (\gamma' \wedge I) \vee \bigvee_{j=1}^{i} f(n_j, \alpha_j, \beta, \gamma'),$$

where $I = h(n, \alpha, \beta)$. In particular, $v_0 = \gamma' \wedge I$. In addition, $\alpha_0 = \alpha$, and $\alpha_i = \alpha \vee v_{i-1}$ for $1 < i < k$.

To prove (6.12) for node n, we need the following lemma:

Lemma B.2.2. *For every $0 \leq i < k$, we have*

$$\beta \wedge (\alpha \vee v_i) = \beta \wedge \left(\alpha \vee \left(\gamma' \wedge \left(I \vee \bigvee_{j=1}^{i} \mathsf{val}(n_j)\right)\right)\right). \tag{B.7}$$

Proof. We prove (B.7) by an induction on i. When $i = 0$, both sides of (B.7) equal $\beta \wedge (\alpha \vee (\gamma' \wedge I))$, hence the equality holds.

For $i \geq 1$, assume that v_{i-1} satisfies (B.7). By induction from the main proposition, $f(n_i, \alpha_i, \beta, \gamma')$ satisfies (6.12), which means

$$\beta \wedge \left(\alpha_i \vee f(n_i, \alpha_i, \beta, \gamma')\right) = \beta \wedge \left(\alpha_i \vee \left(g(n_i, \gamma') \wedge \mathsf{val}(n_i)\right)\right). \tag{B.8}$$

we first establish the following equality

$$g(n_i, \gamma') \wedge \mathsf{val}(n_i) = \gamma' \wedge \mathsf{val}(n_i) \tag{B.9}$$

using the fact that the gamma update function g is admissible.

- If g is node-admissible, then $g(n_i, \gamma') \succeq \mathsf{val}(n_i)$ and $\gamma' = g(n, \gamma) \succeq \mathsf{val}(n) \succeq \mathsf{val}(n_i)$ since n is an OR-node. Hence,

$$g(n_i, \gamma') \wedge \mathsf{val}(n_i) = \mathsf{val}(n_i) = \gamma' \wedge \mathsf{val}(n_i).$$

- If g is path-admissible, then $\gamma' \succeq g(n_i, \gamma') \succeq \gamma' \wedge \mathsf{val}(n_i)$. Hence,

$$\gamma' \wedge \mathsf{val}(n_i) \succeq g(n_i, \gamma') \wedge \mathsf{val}(n_i) \succeq \gamma' \wedge \mathsf{val}(n_i) \wedge \mathsf{val}(n_i) = \gamma' \wedge \mathsf{val}(n_i).$$

[5] By after the 0th loop, we mean before the beginning of the first loop.

In conclusion, (B.9) holds if g is admissible.

Recall that $\alpha_i = \alpha \vee v_{i-1}$, we have

$$\begin{aligned}
\beta \wedge (\alpha \vee v_i) &= \beta \wedge \left(\alpha \vee v_{i-1} \vee f(n_i, \alpha_i, \beta, \gamma')\right) \\
&= \beta \wedge \left(\alpha \vee v_{i-1} \vee \left(g(n_i, \gamma') \wedge \mathsf{val}(n_i)\right)\right) \\
&= \beta \wedge \left(\alpha \vee \left(\gamma' \wedge \left(I \vee \bigvee_{j=1}^{i-1} \mathsf{val}(n_j)\right)\right) \vee \left(g(n_i, \gamma') \wedge \mathsf{val}(n_i)\right)\right) \\
&= \beta \wedge \left(\alpha \vee \left(\gamma' \wedge \left(I \vee \bigvee_{j=1}^{i} \mathsf{val}(n_j)\right)\right)\right),
\end{aligned}$$

where the second line is by distributivity and (B.8), the third line is by distributivity and (B.7) applied to v_{i-1}, and the last line is by distributivity and (B.9). Therefore, (B.7) holds. □

Now we can prove (6.12) for an internal OR-node n. For n with $b \geq 1$ children, two cases are possible:

- No break has taken place, which means $k = b + 1$ and the algorithm has looped through all children of n. Then we have $f(n, \alpha, \beta, \gamma) = v_b$. Plugging $i = b$ into (B.7), we get

  $$\begin{aligned}
  \beta \wedge (\alpha \vee v_b) &= \beta \wedge \left(\alpha \vee \left(\gamma' \wedge \left(I \vee \bigvee_{j=1}^{b} \mathsf{val}(n_j)\right)\right)\right) \\
  &= \beta \wedge \left(\alpha \vee \left(\gamma' \wedge \left(I \vee \mathsf{val}(n)\right)\right)\right) \\
  &= \beta \wedge \left(\alpha \vee \left(\gamma' \wedge \left(\alpha \vee I \vee \mathsf{val}(n)\right)\right)\right) \\
  &= \beta \wedge \left(\alpha \vee \left(\gamma' \wedge \left(\alpha \vee \mathsf{val}(n)\right)\right)\right) \\
  &= \beta \wedge \left(\alpha \vee \left(\gamma' \wedge \mathsf{val}(n)\right)\right)
  \end{aligned}$$

  where the third line and the last line are by insertion identity (A.1) and the fourth line is by $I = h(n, \alpha, \beta) \preceq \mathsf{val}(n) \vee \alpha$ for an OR-node since h is admissible. As a result, (6.12) holds for node n.

- A break happens during the k-th loop, where $1 \leq k \leq b$, which means $\alpha_{k-1} \vee v_{k-1} = \alpha \vee v_{k-1} \succeq \beta$ (Line 11 in Algorithm 6) and $f(n, \alpha, \beta, \gamma) = v_{k-1}$. On the left-hand side of (6.12), by (B.7) applied to $i = k - 1$, we have

  $$\begin{aligned}
  \beta \wedge (\alpha \vee v_{k-1}) &= \beta \wedge \left(\alpha \vee \left(\gamma' \wedge \left(I \vee \bigvee_{j=1}^{k-1} \mathsf{val}(n_j)\right)\right)\right) \\
  &\preceq \beta \wedge \left(\alpha \vee \left(\gamma' \wedge \left(I \vee \bigvee_{j=1}^{b} \mathsf{val}(n_j)\right)\right)\right).
  \end{aligned}$$

Using insertion identity (A.1), we have

$$\begin{aligned}\beta \wedge (\alpha \vee v_{k-1}) &\preceq \beta \wedge \Big(\alpha \vee \big(\gamma' \wedge (I \vee \mathsf{val}(n))\big)\Big) \\ &= \beta \wedge \Big(\alpha \vee \big(\gamma' \wedge (\alpha \vee I \vee \mathsf{val}(n))\big)\Big) \\ &= \beta \wedge \Big(\alpha \vee \big(\gamma' \wedge (\alpha \vee \mathsf{val}(n))\big)\Big) \\ &= \beta \wedge \Big(\alpha \vee \big(\gamma' \wedge \mathsf{val}(n)\big)\Big),\end{aligned}$$

where the third line is by $I \preceq \mathsf{val}(n) \vee \alpha$ (since h is admissible). Hence,

$$\beta = \beta \wedge (\alpha \vee v_{k-1}) \preceq \beta \wedge \Big(\alpha \vee \big(\gamma' \wedge \mathsf{val}(n)\big)\Big) \preceq \beta,$$

which means all inequalities are equalities. As a result, (6.12) holds for node n, and in consequence (6.13) also holds for node n.

Induction for AND-nodes Consider the case in which n is an internal AND-node. Let $b \geq 1$ be the number of children of n and let $n_1, \ldots, n_b$ be the children of n, listed in the same order as in Algorithm 6. By definition, $\mathsf{val}(n) = \bigwedge_{j=1}^{b} \mathsf{val}(n_j)$. Let k be the index of the loop during which a break happens (i.e. a cut is found). If no break happens, then k is taken to be $b + 1$. For $0 \leq i < k$, let v_i and β_i denote the value of v and β after the i-th loop[6]. Then we have

$$v_i = \gamma' \wedge I \wedge \bigwedge_{j=1}^{i} f(n_j, \alpha, \beta_j, \gamma'),$$

where $\gamma' = g(n, \gamma)$. In particular, $v_0 = \gamma' \wedge I$. In addition, $\beta_0 = \beta$, and $\beta_i = \beta \wedge v_{i-1}$ for $1 < i < k$.

To prove (6.13) (which is equivalent to (6.12)) for node n, we need the following lemma:

Lemma B.2.3. *For every $0 \leq i < k$, we have*

$$\alpha \vee (\beta \wedge v_i) = \alpha \vee \left(\beta \wedge \gamma' \wedge I \wedge \bigwedge_{j=1}^{i} \mathsf{val}(n_j)\right). \tag{B.10}$$

Proof. We prove (B.10) by an induction on i. When $i = 0$, both sides of (B.10) equal $\alpha \vee (\beta \wedge \gamma' \wedge I)$, hence the equality holds.

For $i \geq 1$, assume that v_{i-1} satisfies (B.10). By induction from the main proposition, $f(n_i, \alpha, \beta_i, \gamma')$ satisfies (6.13), which means

$$\alpha \vee \big(\beta_i \wedge f(n_i, \alpha, \beta_i, \gamma')\big) = \alpha \vee \Big(\beta_i \wedge \big(g(n_i, \gamma') \wedge \mathsf{val}(n_i)\big)\Big). \tag{B.11}$$

We first establish the following equality

$$\gamma' \wedge g(n_i, \gamma') \wedge \mathsf{val}(n_i) = \gamma' \wedge \mathsf{val}(n_i) \tag{B.12}$$

using the fact that the gamma update function g is admissible.

[6] By after the 0th loop, we mean before the beginning of the first loop.

- If g is node-admissible, then $g(n_i, \gamma') \succeq \mathsf{val}(n_i)$ hence (B.12) holds.
- If g is path-admissible, then $g(n_i, \gamma') \succeq \gamma' \wedge \mathsf{val}(n_i)$ hence (B.12) holds.

In conclusion, (B.12) holds if g is admissible.

Since $\beta_i = \beta \wedge v_{i-1}$, we have

$$\begin{aligned}
\alpha \vee (\beta \wedge v_i) &= \alpha \vee \big(\beta \wedge v_{i-1} \wedge f(n_i, \alpha, \beta_i, \gamma')\big) \\
&= \alpha \vee \big(\beta \wedge v_{i-1} \wedge g(n_i, \gamma') \wedge \mathsf{val}(n_i)\big) \\
&= \alpha \vee \left(\beta \wedge \gamma' \wedge I \wedge \bigwedge_{j=1}^{i-1} \mathsf{val}(n_j) \wedge g(n_i, \gamma') \wedge \mathsf{val}(n_i)\right) \\
&= \alpha \vee \left(\beta \wedge \gamma' \wedge I \wedge \bigwedge_{j=1}^{i} \mathsf{val}(n_j)\right),
\end{aligned}$$

where the second line is by distributivity and (B.11), the third line is by distributivity and (B.10) applied to v_{i-1}, and the last line is by (B.12). Therefore, (B.10) holds. □

Now we can prove (6.13) for an internal AND-node n. For n with $b \geq 1$ children, two cases are possible:

- No break has taken place, which means $k = b + 1$ and the algorithm has looped through all children of n. Then we have $f(n, \alpha, \beta, \gamma) = v_b$. Plugging $i = b$ into (B.10), we get

  $$\begin{aligned}
  \alpha \vee (\beta \wedge v_b) &= \alpha \vee \left(\beta \wedge \gamma' \wedge I \wedge \bigwedge_{j=1}^{b} \mathsf{val}(n_j)\right) \\
  &= \alpha \vee \big(\beta \wedge \gamma' \wedge I \wedge \mathsf{val}(n)\big) \\
  &= \alpha \vee \big(\beta \wedge \gamma' \wedge \mathsf{val}(n)\big)
  \end{aligned}$$

  where the third line is by $I = h(n, \alpha, \beta) \succeq \mathsf{val}(n) \wedge \beta$ for an AND-node since h is admissible. As a result, (6.13) holds for node n.

- A break happens during the k-th loop, where $1 \leq k \leq b$, which means $\alpha \succeq \beta_{k-1} \wedge v_{k-1} = \beta \wedge v_{k-1}$ (Line 11 in Algorithm 6) and $f(n, \alpha, \beta, \gamma) = v_{k-1}$. On the left-hand side of (6.13), we have

  $$\begin{aligned}
  \alpha \vee (\beta \wedge v_{k-1}) &= \alpha \vee \left(\beta \wedge \gamma' \wedge I \wedge \bigwedge_{j=1}^{k-1} \mathsf{val}(n_j)\right) \\
  &\succeq \alpha \vee \left(\beta \wedge \gamma' \wedge I \wedge \bigwedge_{j=1}^{b} \mathsf{val}(n_j)\right) \\
  &= \alpha \vee \big(\beta \wedge \gamma' \wedge I \wedge \mathsf{val}(n)\big) \\
  &= \alpha \vee \big(\beta \wedge \gamma' \wedge \mathsf{val}(n)\big),
  \end{aligned}$$

  where the first line is by (B.10) applied to $i = k - 1$, and the last line is by $I \succeq \mathsf{val}(n) \wedge \beta$ (since h is admissible). Hence,

  $$\alpha = \alpha \vee (\beta \wedge v_{k-1}) \succeq \alpha \vee \big(\beta \wedge \gamma' \wedge \mathsf{val}(n)\big) \succeq \alpha,$$

  which means all inequalities are equalities. As a result, (6.13) holds for node n, and in consequence (6.12) also holds for node n.

Conclusion We have therefore established by induction that (6.12) and (6.13) hold for all nodes n of G. □

Remark. *The admissibility of g is only used to prove* (B.9) *and* (B.12)*. More concretely, what we need is that the following equations hold:*

- *if n' is a child of an OR-node n, then*

$$\forall \gamma \in V, g(n', g(n, \gamma)) \wedge \mathsf{val}(n') = g(n, \gamma) \wedge \mathsf{val}(n');$$

- *if n' is a child of an AND-node n, then*

$$\forall \gamma \in V, g(n, \gamma) \wedge g(n', g(n, \gamma)) \wedge \mathsf{val}(n') = g(n, \gamma) \wedge \mathsf{val}(n').$$

Notice that the requirement on a child of an OR-node is stricter than on a child of an AND-node. We have shown that both node-admissibility and path-admissibility are sufficient to imply (B.9) *and* (B.12)*, but not necessary. This means one may also define less strict notions of admissibility while maintaining the correctness of Proposition 6.5.2, which would allow richer application of alpha-beta-gamma prunings.*

B.3 Proofs for Chapter 7

Throughout this section, we write $G \coloneqq \langle T, \mathrm{P}, t, \vec{u}, \vec{\rho}\rangle$ for a CGII. Recall that the NBS at node n about OM ω_-, written as $\mathrm{NBS}(n, \omega_-) \in [0,1]^t$, is defined by $\mathrm{NBS}(r, \omega_-) \coloneqq \vec{\rho}$ and for $n' \in \mathrm{C}(n)$, if $n \in N_+$ then $\mathrm{NBS}(n', \omega_-) \coloneqq \mathrm{NBS}(n, \omega_-)$, otherwise

$$\mathrm{NBS}(n', \omega_-)_i \coloneqq \mathrm{NBS}(n, \omega_-)_i \times \omega_-(n, i, n').$$

Payoff of any subtree under a profile Let s_+ and s_- be any pure strategy of MAX and MIN, respectively. For every type i of MIN, let n_i be the leaf reached if MAX and MIN play respectively s_+ and s_-, and the game begins at n. We define the vector

$$\vec{u}(n, s_+, s_-) \coloneqq \big(\vec{u}(n_1)_1, \ldots, \vec{u}(n_t)_t\big),$$

which is interpreted as the payoff vector if MAX and MIN play respectively s_+ and s_-, and the game begins at n. In particular, $\mathcal{U}(s_+, s_-) = \vec{\rho} \cdot \vec{u}(r, s_+, s_-)$. For mixed strategies σ_+ and σ_- of MAX and MIN, respectively, $\vec{u}(n, \sigma_+, \sigma_-)$ is the expectation of $\vec{u}(n, s_+, s_-)$ when s_+ and s_- are drawn according to σ_+ and σ_-, respectively.

Notice that $\vec{u}(n, s_+, s_-)$ only depends on the choices of strategy s_+ at nodes in the subtree rooted at n. More precisely, if s_+ and s'_+ choose the same action at every node in the subtree rooted at n, then $\vec{u}(n, s_+, s_-) = \vec{u}(n, s'_+, s_-)$. In addition, for every MAX's node n and $s_- \in \Sigma_-^{\mathrm{P}}$, we have

$$\big\{\vec{u}(n, s_+, s_-)\big|\, s_+ \in \Sigma_+^{\mathrm{P}}\big\} = \bigcup_{n' \in \mathrm{C}(n)} \big\{\vec{u}(n', s_+, s_-)\big|\, s_+ \in \Sigma_+^{\mathrm{P}}\big\}. \tag{B.13}$$

Proposition 7.3.2. *Let $\langle T, \mathrm{P}, t, \vec{u}, \vec{\rho}\rangle$ be a vector game with root r, and let ω_- be an OM. For $l \in \mathrm{L}(T)$, let $\mathsf{eval}(l) \coloneqq \mathrm{NBS}(l, \omega_-) \cdot \vec{u}(l)$. Then $\texttt{MiniMax}(\mathbb{R}, \mathsf{eval}, \max, +)$ satisfies*

$$\underline{v}_+ \coloneqq \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \mathcal{U}(s_+, \omega_-) = \max_{\sigma_+ \in \Sigma_+^{\mathrm{M}}} \mathcal{U}(\sigma_+, \omega_-) = \mathsf{val}(r),$$

and runs in polynomial time (more precisely, in $O(t|T|)$ time).[7]

Proof. We only consider MAX's pure strategies in the following. We will establish by induction on the game tree that for every node n, its situational value $\mathsf{val}(n)$ satisfies

$$\mathsf{val}(n) = \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \mathrm{NBS}(n, \omega_-) \cdot \vec{u}(n, s_+, \omega_-). \tag{B.14}$$

First, consider a leaf n. Then

$$\mathsf{val}(n) = \mathsf{eval}(n) \coloneqq \mathrm{NBS}(n, \omega_-) \cdot \vec{u}(n).$$

Since n is a leaf, by definition of $\vec{u}(n, s_+, \omega_-)$, it trivially holds that $\vec{u}(n, s_+, \omega_-) = \vec{u}(n)$ for all $s_+ \in \Sigma_+^{\mathrm{P}}$. Hence, (B.14) holds for n.

Now consider an internal node n. If n is MAX's decision node, then $\mathsf{val}(n) = \max_{n' \in \mathrm{C}(n)} \mathsf{val}(n')$. Applying the induction hypothesis to n' yields

$$\mathsf{val}(n) = \max_{n' \in \mathrm{C}(n)} \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \mathrm{NBS}(n', \omega_-) \cdot \vec{u}(n', s_+, \omega_-),$$

[7] The proof of this and other results in this chapter can be found in the appendix.

from which we deduce that (B.14) holds for n using (B.13) and the fact that for all $n' \in \mathrm{C}(n)$, $\mathrm{NBS}(n', \omega_-) = \mathrm{NBS}(n, \omega_-)$.

If n is MIN's decision node, then we have $\mathsf{val}(n) = \sum_{n' \in \mathrm{C}(n)} \mathsf{val}(n')$. Applying the induction hypothesis to n' yields

$$\mathsf{val}(n) = \sum_{n' \in \mathrm{C}(n)} \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \mathrm{NBS}(n', \omega_-) \cdot \vec{u}(n', s_+, \omega_-).$$

Now, since each n' is a different node in the game tree, MAX can apply any combination of strategies in the subtree rooted at these nodes. Hence, the sum over n' can be exchanged with the maximum, which yields

$$\mathsf{val}(n) = \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \sum_{n' \in \mathrm{C}(n)} \mathrm{NBS}(n', \omega_-) \cdot \vec{u}(n', s_+, \omega_-).$$

Finally, by definition of $\vec{u}$ and NBS, $\mathrm{NBS}(n, \omega_-) \cdot \vec{u}(n, s_+, \omega_-)$ can be written as

$$\sum_{n' \in \mathrm{C}(n)} \mathrm{NBS}(n', \omega_-) \cdot \vec{u}(n', s_+, \omega_-),$$

since n is MIN's node, and MIN chooses n' according to ω_-. Therefore, (B.14) holds for n, which concludes the induction.

Applying (B.14) to the root, we have

$$\begin{aligned} \mathsf{val}(r) &= \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \mathrm{NBS}(r, \omega_-) \cdot \vec{u}(r, s_+, \omega_-) \\ &= \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \mathcal{U}(s_+, \omega_-), \end{aligned}$$

where we use the fact that $\mathrm{NBS}(r, \omega_-) = \vec{\rho}$ and $\mathcal{U}(s_+, \omega_-) = \vec{\rho} \cdot \vec{u}(r, s_+, \omega_-)$. Therefore, the situational value of the root is the maxmin value. □

Proposition 7.3.3. *Let $\langle T, \mathrm{P}, t, \vec{u}, \vec{\rho} \rangle$ be a vector game with root r, and let $\omega_-^1, \ldots, \omega_-^m$ be OMs distributed according to $\vec{p} = (p_1, \ldots, p_m)$. Let $\mathsf{eval}(l) \coloneqq \sum_{j=1}^m p_j \, \mathrm{NBS}(l, \omega_-^j) \cdot \vec{u}(l)$ for $l \in \mathrm{L}(T)$. Then $\texttt{MiniMax}(\mathbb{R}, \mathsf{eval}, \max, +)$ satisfies $\underline{v}_+ = \mathsf{val}(r)$ and runs in polynomial time (more precisely, in $O(mt|T|)$ time).*

Proof. Recall that the maxmin value under this setting is defined to be

$$\underline{v}_+ \coloneqq \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \sum_{j=1}^m p_j \mathcal{U}(s_+, \omega_-^j).$$

Also, recall that MIN has perfect information in a vector game, and *a fortiori* perfect recall. Hence, every MIN's behaviour strategy has an equivalent mixed strategy, and vice versa. Since $\sum_{j=1}^m p_j = 1$, if we interpret the OMs $(\omega_-^j)_{1 \le j \le m}$ as mixed strategies of MIN, then

$$\omega_- \coloneqq p_1 \omega_-^1 + \cdots + p_m \omega_-^m$$

is also a probability distribution over pure strategies of MIN and is thus a well-defined mixed strategy of MIN.

MAX's NBS corresponding to ω_- satisfies

$$\mathrm{NBS}(n, \omega_-) = \sum_{j=1}^m p_j \cdot \mathrm{NBS}(n, \omega_-^j)$$

for every node n. This can be verified using the recursive definition of NBS by noticing that a behaviour strategy ω_-^{b} equivalent to ω_- satisfies for all nodes n, types i, and nodes $n' \in \mathrm{C}(n)$:

$$\omega_-^{\mathrm{b}}(n, i, n') = \frac{\sum_{j=1}^m p_j \prod_{(n_1,n_2)\in \mathrm{Path}(n')} \omega_-^j(n_1, i, n_2)}{\sum_{j=1}^m p_j \prod_{(n_1,n_2)\in \mathrm{Path}(n)} \omega_-^j(n_1, i, n_2)}$$

where $\mathrm{Path}(n)$ (respectively $\mathrm{Path}(n')$) denotes the set of all edges (n_1, n_2) in the path from r to n (respectively n').

Hence, the function eval reads

$$\mathsf{eval}(n) \coloneqq \sum_{j=1}^m p_j \,\mathrm{NBS}(n, \omega_-^j) \cdot \vec{u}(n) = \mathrm{NBS}(n, \omega_-) \cdot \vec{u}(n)$$

for all leaf nodes n. Then by Proposition 7.3.2, $\mathsf{val}(r)$ is the maxmin value against the OM ω_-, which reads

$$\mathsf{val}(r) = \max_{s_+\in\Sigma_+^{\mathrm{P}}} \mathcal{U}(s_+, \omega_-) = \max_{s_+\in\Sigma_+^{\mathrm{P}}} \sum_{j=1}^m p_j \mathcal{U}(s_+, \omega_-^j),$$

where the second equality is due to the linearity of $\mathcal{U}$. Hence, $\mathsf{val}(r)$ is exactly the maxmin value against the probabilistic OMs. □

Proposition 7.3.4. *Let $\langle T, \mathrm{P}, t, \vec{u}, \vec{\rho}\rangle$ be a vector game with root r, and let $\omega_-^1, \ldots, \omega_-^m$ be OMs with a lexicographic interpretation. Let $\mathsf{eval}(l) \coloneqq \mathrm{NBSM}(l) \times \vec{u}(l) \in \mathbb{R}^m$ for $l \in \mathrm{L}(T)$. Then $\mathtt{MiniMax}(\mathbb{R}^m, \mathsf{eval}, \mathrm{lexmax}, +^m)$ satisfies $\overrightarrow{v_+} = \mathsf{val}(r)$ and runs in polynomial time (more precisely, in $O(mt|T|)$ time).*

Proof. In the following, we write a vector $(a_1, \ldots, a_m)$ of length m as $(a_j)_{1\le j\le m}$ to save space. For example, the definition of eval reads

$$\mathsf{eval}(l) \coloneqq \left(\mathrm{NBS}(l, \omega_-^j) \cdot \vec{u}(l)\right)_{1\le j\le m}.$$

Recall that the maxmin value under this setting is defined to be

$$\overrightarrow{v_+} \coloneqq \operatorname*{lexmax}_{s_+\in\Sigma_+^{\mathrm{P}}} \left(\mathcal{U}(s_+, \omega_-^j)\right)_{1\le j\le m} \in \mathbb{R}^m.$$

The proof of Proposition 7.3.4 is nearly identical to the one of Proposition 7.3.2. The only difference is that situational values are now real vectors of length m instead of real numbers. We will establish by induction on the game tree that for every node n, its situational value $\mathsf{val}(n) \in \mathbb{R}^m$ satisfies

$$\mathsf{val}(n) = \operatorname*{lexmax}_{s_+\in\Sigma_+^{\mathrm{P}}} \left(\mathrm{NBS}(n, \omega_-^j) \cdot \vec{u}(n, s_+, \omega_-^j)\right)_{1\le j\le m}. \tag{B.15}$$

First, consider a leaf n. Then

$$\mathsf{val}(n) = \mathsf{eval}(n) \coloneqq \left(\mathrm{NBS}(n, \omega_-^j) \cdot \vec{u}(n)\right)_{1\le j\le m}.$$

Since n is a leaf, it trivially holds that $\vec{u}(n, s_+, \omega_-^j) = \vec{u}(n)$ for all $s_+ \in \Sigma_+^{\mathrm{P}}$ and $1 \le j \le m$. Hence, (B.15) holds for n.

Now consider an internal node n. If n is MAX's decision node, then $\mathsf{val}(n) = \operatorname{lexmax}_{n' \in \mathrm{C}(n)} \mathsf{val}(n')$. Applying the induction hypothesis to n', $\mathsf{val}(n)$ reads

$$\operatorname*{lexmax}_{n' \in \mathrm{C}(n)} \operatorname*{lexmax}_{s_+ \in \Sigma_+^{\mathrm{P}}} \left(\mathrm{NBS}(n', \omega_-^j) \cdot \vec{u}(n', s_+, \omega_-^j) \right)_{1 \le j \le m},$$

from which we deduce that (B.15) holds for n using (B.13) and the fact that for all $n' \in \mathrm{C}(n)$ and $1 \le j \le m$, $\mathrm{NBS}(n', \omega_-^j) = \mathrm{NBS}(n, \omega_-^j)$.

If n is MIN's decision node, then we have $\mathsf{val}(n) = \sum_{n' \in \mathrm{C}(n)} \mathsf{val}(n')$. Applying the induction hypothesis to n', $\mathsf{val}(n)$ reads

$$\sum_{n' \in \mathrm{C}(n)} \operatorname*{lexmax}_{s_+ \in \Sigma_+^{\mathrm{P}}} \left(\mathrm{NBS}(n', \omega_-^j) \cdot \vec{u}(n', s_+, \omega_-^j) \right)_{1 \le j \le m}.$$

Now, since each n' is a different node in the game tree, MAX can apply any combination of strategies in the subtree rooted at these nodes. Hence, the sum over n' can be exchanged with the maximum, which means $\mathsf{val}(n)$ reads

$$\operatorname*{lexmax}_{s_+ \in \Sigma_+^{\mathrm{P}}} \sum_{n' \in \mathrm{C}(n)} \left(\mathrm{NBS}(n', \omega_-^j) \cdot \vec{u}(n', s_+, \omega_-^j) \right)_{1 \le j \le m}.$$

Finally, for all $1 \le j \le m$, $\mathrm{NBS}(n, \omega_-^j) \cdot \vec{u}(n, s_+, \omega_-^j)$ can be written as

$$\sum_{n' \in \mathrm{C}(n)} \mathrm{NBS}(n', \omega_-^j) \cdot \vec{u}(n', s_+, \omega_-^j),$$

since n is MIN's node, and MIN under the j-th OM chooses n' according to ω_-^j. Therefore, (B.15) holds for n, which concludes the induction.

Applying (B.15) to the root, we have

$$\mathsf{val}(r) = \operatorname*{lexmax}_{s_+ \in \Sigma_+^{\mathrm{P}}} \left(\mathrm{NBS}(r, \omega_-^j) \cdot \vec{u}(r, s_+, \omega_-^j) \right)_{1 \le j \le m}.$$

Using the fact that for all $1 \le j \le m$, $\mathrm{NBS}(r, \omega_-^j) = \vec{\rho}$ and $\mathcal{U}(s_+, \omega_-^j) = \vec{\rho} \cdot \vec{u}(r, s_+, \omega_-^j)$, we get $\mathsf{val}(r) = \overrightarrow{\underline{v}_+}$. □

Proposition 7.3.5. *Let $\langle T, \mathrm{P}, t, \vec{u}, \vec{\rho} \rangle$ be a game with root r, and let $\omega_-^1, \ldots, \omega_-^m$ be OMs with a nondeterministic interpretation. For $l \in \mathrm{L}(T)$, let $\mathsf{eval}(l) \coloneqq \{\mathrm{NBSM}(l) \times \vec{u}(l)\}$. Then $\mathtt{MiniMax}(\mathcal{P}_{<\infty}(\mathbb{R}^m), \mathsf{eval}, \cup, \oplus^m)$ satisfies*

$$\underline{v}_+ \coloneqq \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \min_{1 \le j \le m} \mathcal{U}(s_+, \omega_-^j) = \max_{\vec{v} \in \mathsf{val}(r)} \min_{1 \le j \le m} v_j.$$

Proof. We will establish by induction on the game tree that for every node n, its situational value $\mathsf{val}(n)$ satisfies

$$\mathsf{val}(n) = \left\{ \left(\mathrm{NBS}(n, \omega_-^j) \cdot \vec{u}(n, s_+, \omega_-^j) \right)_{1 \le j \le m} \,\middle|\, s_+ \in \Sigma_+^{\mathrm{P}} \right\}. \tag{B.16}$$

In other words, $\mathtt{MiniMax}(\mathcal{P}_{<\infty}(\mathbb{R}^m), \mathsf{eval}, \cup, \oplus^m)$ implicitly and recursively enumerates all pure strategies of MAX.

First, consider a leaf n. Then

$$\mathsf{val}(n) = \mathsf{eval}(n) \coloneqq \left\{ \left(\mathrm{NBS}(n, \omega_-^j) \cdot \vec{u}(n) \right)_{1 \le j \le m} \right\}.$$

Since n is a leaf, it trivially holds that $\vec{u}(n, s_+, \omega_-^j) = \vec{u}(n)$ for all $s_+ \in \Sigma_+^{\mathrm{P}}$ and $1 \leq j \leq m$. Hence, (B.16) holds for n.

Now consider an internal node n. If n is MAX's decision node, then $\mathsf{val}(n) = \cup_{n' \in \mathrm{C}(n)} \mathsf{val}(n')$. Applying the induction hypothesis to n', $\mathsf{val}(n)$ reads

$$\bigcup_{n' \in \mathrm{C}(n)} \left\{ \left(\mathrm{NBS}(n', \omega_-^j) \cdot \vec{u}(n', s_+, \omega_-^j) \right)_{1 \leq j \leq m} \middle| \, s_+ \in \Sigma_+^{\mathrm{P}} \right\},$$

from which we deduce that (B.16) holds for n using (B.13) and the fact that for all $n' \in \mathrm{C}(n)$ and $1 \leq j \leq m$, $\mathrm{NBS}(n', \omega_-^j) = \mathrm{NBS}(n, \omega_-^j)$.

If n is MIN's decision node, then we have $\mathsf{val}(n) = \oplus^m_{n' \in \mathrm{C}(n)} \mathsf{val}(n')$, where for $f, g \in \mathcal{P}_{<\infty}(\mathbb{R}^m)$, $\oplus^m$ is defined by

$$f \oplus^m g \coloneqq \{(v_j + v'_j)_{1 \leq j \leq m} \mid \vec{v} \in f, \vec{v}\,' \in g\} \subseteq \mathbb{R}^m.$$

Applying the induction hypothesis to n', $\mathsf{val}(n)$ reads

$$\bigoplus_{n' \in \mathrm{C}(n)}^m \left\{ \left(\mathrm{NBS}(n', \omega_-^j) \cdot \vec{u}(n', s_+, \omega_-^j) \right)_{1 \leq j \leq m} \middle| \, s_+ \in \Sigma_+^{\mathrm{P}} \right\}.$$

For all $1 \leq j \leq m$, $\mathrm{NBS}(n, \omega_-^j) \cdot \vec{u}(n, s_+, \omega_-^j)$ can be written as

$$\sum_{n' \in \mathrm{C}(n)} \mathrm{NBS}(n', \omega_-^j) \cdot \vec{u}(n', s_+, \omega_-^j),$$

since n is MIN's node, and MIN under the j-th OM chooses n' according to ω_-^j. Therefore, (B.16) holds for n, which concludes the induction.

Applying (B.16) to the root, we have

$$\mathsf{val}(r) = \left\{ \left(\mathrm{NBS}(r, \omega_-^j) \cdot \vec{u}(r, s_+, \omega_-^j) \right)_{1 \leq j \leq m} \middle| \, s_+ \in \Sigma_+^{\mathrm{P}} \right\}.$$

Using the fact that for all $1 \leq j \leq m$, $\mathrm{NBS}(r, \omega_-^j) = \vec{\rho}$ and $\mathcal{U}(s_+, \omega_-^j) = \vec{\rho} \cdot \vec{u}(r, s_+, \omega_-^j)$, we get

$$\mathsf{val}(r) = \left\{ \left(\mathcal{U}(s_+, \omega_-^1), \ldots, \mathcal{U}(s_+, \omega_-^m) \right) \middle| \, s_+ \in \Sigma_+^{\mathrm{P}} \right\}.$$

Therefore,

$$\underline{v}_+ \coloneqq \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \min_{1 \leq j \leq m} \mathcal{U}(s_+, \omega_-^j) = \max_{\vec{v} \in \mathsf{val}(r)} \min_{1 \leq j \leq m} v_j.$$

□

Proposition 7.3.6. *Let $\langle T, \mathrm{P}, t, \vec{u}, \vec{\rho} \rangle$ be a vector game with root r, ω_- be an OM, and $p^\infty \in [0, 1]$ a probability that MIN does not follow ω_-. For all $l \in \mathrm{L}(T)$, let*

$$\mathsf{eval}(l) \coloneqq \{\langle \mathrm{NBS}(l, \omega_-) \cdot \vec{u}(l), \vec{u}(l) \rangle\} \in \mathcal{P}_{<\infty}(\mathbb{R} \times \mathbb{R}^t).$$

Then $\mathtt{MiniMax}(\mathcal{P}_{<\infty}(\mathbb{R} \times \mathbb{R}^t), \mathsf{eval}, \cup, \oplus^{1,t})$ *satisfies*

$$\underline{v}_+ = \max_{\langle s, \vec{v} \rangle \in \mathsf{val}(r)} \left((1 - p^\infty) s + p^\infty (\vec{q} \cdot \vec{v}) \right).$$

Proof. Recall that the maxmin value under this setting is defined to be

$$\underline{v}_+ \coloneqq \max_{s_+ \in \Sigma_+^{\mathrm{P}}} \left((1 - p^\infty) \mathcal{U}(s_+, \omega_-) + p^\infty \min_{s_- \in \Sigma_-^{\mathrm{P}}} \mathcal{U}(s_+, s_-) \right).$$

We will establish by induction on the game tree that for every node n, its situational value $\mathsf{val}(n)$ satisfies

$$\mathsf{val}(n) = \left\{ \left\langle \mathrm{NBS}(n, \omega_-) \cdot \vec{u}(n, s_+, \omega_-), \min_{s_- \in \Sigma_-^{\mathrm{P}}} \vec{u}(n, s_+, s_-) \right\rangle \middle| \, s_+ \in \Sigma_+^{\mathrm{P}} \right\}, \tag{B.17}$$

where min is the component-wise minimum of vectors of length m. In other words, the situational value of n stores, for each pure strategy of MAX, the payoff of this pure strategy under the subtree rooted at n against the OM ω_-, and its worst payoff against each type of MIN.

First, consider a leaf n. Then

$$\mathsf{val}(n) = \mathsf{eval}(n) \coloneqq \{\langle \mathrm{NBS}(n, \omega_-) \cdot \vec{u}(n), \vec{u}(n) \rangle\}.$$

Since n is a leaf, it trivially holds that $\vec{u}(n, s_+, s_-) = \vec{u}(n)$ for all $s_+ \in \Sigma_+^{\mathrm{P}}$ and all $s_- \in \Sigma_-^{\mathrm{P}}$. In addition, $\vec{u}(n, s_+, \omega_-) = \vec{u}(n)$, Hence, (B.17) holds for n.

Now consider an internal node n. If n is MAX's decision node, then $\mathsf{val}(n) = \cup_{n' \in \mathrm{C}(n)} \mathsf{val}(n')$. Applying the induction hypothesis to n' yields

$$\mathsf{val}(n) = \bigcup_{n' \in \mathrm{C}(n)} \left\{ \left\langle \mathrm{NBS}(n', \omega_-) \cdot \vec{u}(n', s_+, \omega_-), \min_{s_- \in \Sigma_-^{\mathrm{P}}} \vec{u}(n', s_+, s_-) \right\rangle \middle| \, s_+ \in \Sigma_+^{\mathrm{P}} \right\},$$

from which we deduce that (B.17) holds for n using (B.13) and the fact that for all $n' \in \mathrm{C}(n)$, $\mathrm{NBS}(n', \omega_-) = \mathrm{NBS}(n, \omega_-)$.

If n is MIN's decision node, then we have $\mathsf{val}(n) = \oplus^{1,t}_{n' \in \mathrm{C}(n)} \mathsf{val}(n')$, where for $f, g \in \mathcal{P}_{<\infty}(\mathbb{R} \times \mathbb{R}^t)$, we define $f \oplus^{1,t} g \subseteq \mathbb{R} \times \mathbb{R}^t$ to be the set

$$\{\langle s + s', (\min(v_i, v_i'))_{1 \le i \le t} \rangle \mid \langle s, \vec{v} \rangle \in f, \langle s', \vec{v}\,' \rangle \in g\}.$$

Applying the induction hypothesis to n' yields

$$\mathsf{val}(n) = \bigoplus_{n' \in \mathrm{C}(n)}^{1,t} \left\{ \left\langle \mathrm{NBS}(n', \omega_-) \cdot \vec{u}(n', s_+, \omega_-), \min_{s_- \in \Sigma_-^{\mathrm{P}}} \vec{u}(n', s_+, s_-) \right\rangle \middle| \, s_+ \in \Sigma_+^{\mathrm{P}} \right\}.$$

Observe that

$$\sum_{n' \in \mathrm{C}(n)} \mathrm{NBS}(n', \omega_-) \cdot \vec{u}(n', s_+, \omega_-) = \mathrm{NBS}(n, \omega_-) \cdot \vec{u}(n, s_+, \omega_-)$$

and

$$\min_{s_- \in \Sigma_-^{\mathrm{P}}} \vec{u}(n, s_+, s_-) = \min_{n' \in \mathrm{C}(n)} \min_{s_- \in \Sigma_-^{\mathrm{P}}} \vec{u}(n', s_+, s_-)$$

since MIN can choose a successor n' of n according to their type. Hence, (B.17) holds for n, which concludes the induction.

Applying (B.17) to the root, and using $\mathrm{NBS}(r, \omega_-) = \vec{\rho}$ and $\mathcal{U}(s_+, \omega_-) = \vec{\rho} \cdot \vec{u}(r, s_+, \omega_-)$, we get

$$\mathsf{val}(r) = \left\{ \left\langle \mathcal{U}(s_+, \omega_-), \min_{s_- \in \Sigma_-^{\mathrm{P}}} \vec{u}(s_+, s_-) \right\rangle \middle| \, s_+ \in \Sigma_+^{\mathrm{P}} \right\}.$$

Therefore,

$$\max_{\langle s, \vec{v} \rangle \in \mathsf{val}(r)} \left((1 - p^\infty) s + p^\infty (\vec{\rho} \cdot \vec{v}) \right) = \underline{v}_+.$$

□

Bibliography

Albert, M. H. and Nowakowski, R. J., editors (2009). *Games of No Chance 3*, volume 56 of *Mathematical Sciences Research Institute Publications*. Cambridge University Press. p10

Albert, M. H., Nowakowski, R. J., and Wolfe, D. (2019). *Lessons in Play: An Introduction to Combinatorial Game Theory*. CRC Press, 2nd edition. p9

Albrecht, S. V. and Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artif. Intell.*, 258:66–95. p13

Allis, L. V., van der Meulen, M., and van den Herik, H. J. (1994). Proof-number search. *Artif. Intell.*, 66(1):91–124. p11

Apt, K. R. and Grädel, E., editors (2011). *Lectures in Game Theory for Computer Scientists*. Cambridge University Press. p6

Arora, S. and Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press. p10, p162

Atzmon, D., Stern, R., and Saffidine, A. (2018). Bounded suboptimal game tree search. In Bulitko, V. and Storandt, S., editors, *Proceedings of the Eleventh International Symposium on Combinatorial Search, SOCS 2018, Stockholm, Sweden - 14-15 July 2018*, pages 10–18. AAAI Press. p119

Babai, L., Fortnow, L., and Lund, C. (1991). Non-deterministic exponential time has two-prover interactive protocols. *Comput. Complex.*, 1:3–40. p30

Balcázar, J. L., Lozano, A., and Torán, J. (1992). *The Complexity of Algorithmic Problems on Succinct Instances*, pages 351–377. Springer US, Boston, MA. p50

Ballard, B. W. (1983). The *-minimax search procedure for trees containing chance nodes. *Artif. Intell.*, 21(3):327–350. p39

Basilico, N., Celli, A., Nittis, G. D., and Gatti, N. (2017). Team-maxmin equilibrium: Efficiency bounds and algorithms. In Singh, S. and Markovitch, S., editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 356–362. AAAI Press. p9, p28, p29

Berlekamp, E. R., Conway, J. H., and Guy, R. K. (2001). *Winning Ways for Your Mathematical Plays, Volume 1*. CRC Press, 2nd edition. p9

Berlekamp, E. R., Conway, J. H., and Guy, R. K. (2003a). *Winning Ways for Your Mathematical Plays, Volume 2*. CRC Press, 2nd edition. p9

Berlekamp, E. R., Conway, J. H., and Guy, R. K. (2003b). *Winning Ways for Your Mathematical Plays, Volume 3*. CRC Press, 2nd edition. p9

Berlekamp, E. R., Conway, J. H., and Guy, R. K. (2004). *Winning Ways for Your Mathematical Plays, Volume 4*. CRC Press, 2nd edition. p9

Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Math. Oper. Res.*, 27(4):819–840. p11, p30

Bethe, P. M. (2021). *Advances in computer bridge: techniques for a partial-information, communication-based game*. PhD thesis, New York University, USA. p12

Bonanno, G. (2018). *Game Theory*. Kindle Direct Publishing. p7, p12

Bonnet, E., Jamain, F., and Saffidine, A. (2013). On the complexity of trick-taking card games. In Rossi, F., editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 482–488. IJCAI/AAAI. p11

Bonnet, É. and Saffidine, A. (2014). On the complexity of general game playing. In Cazenave, T., Winands, M. H. M., and Björnsson, Y., editors, *Computer Games - Third Workshop on Computer Games, CGW 2014, Held in Conjunction with the 21st European Conference on Artificial Intelligence, ECAI 2014, Prague, Czech Republic, August 18, 2014, Revised Selected Papers*, volume 504 of *Communications in Computer and Information Science*, pages 90–104. Springer. p49

Bosanský, B., Kiekintveld, C., Lisý, V., and Pechoucek, M. (2014). An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *J. Artif. Intell. Res.*, 51:829–866. p10

Bourke, T. (2005). *Countdown to Winning Bridge*. Master Point Press. p168

Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In Shoham, Y., editor, *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge, De Zeeuwse Stromen, The Netherlands, March 17-20 1996*, pages 195–210. Morgan Kaufmann. p6

Bowling, M., Burch, N., Johanson, M., and Tammelin, O. (2017). Heads-up limit hold'em poker is solved. *Commun. ACM*, 60(11):81–88. p12

Brafman, R. I., Shani, G., and Zilberstein, S. (2013). Qualitative planning under partial observability in multi-agent domains. In desJardins, M. and Littman, M. L., editors, *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. AAAI Press. p6, p11, p30, p108

Brandt, F., Conitzer, V., Endriss, U., Lang, J., and Procaccia, A. D., editors (2016). *Handbook of Computational Social Choice*. Cambridge University Press. p7

Brown, G. W. (1951). Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 13(1):374–376. p153

Brown, N. and Sandholm, T. (2017). Libratus: The superhuman AI for no-limit poker. In Sierra, C., editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 5226–5228. ijcai.org. p12

Brown, N. and Sandholm, T. (2019). Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890. p12

Browne, C., Powley, E. J., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Liebana, D. P., Samothrakis, S., and Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intell. AI Games*, 4(1):1–43. p11

Buro, M., Long, J. R., Furtak, T., and Sturtevant, N. R. (2009). Improving state evaluation, inference, and search in trick-based card games. In Boutilier, C., editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 1407–1413. p12, p112

Camerer, C. F., Ho, T.-H., and Chong, J.-K. (2004). A cognitive hierarchy model of games. *The Quarterly Journal of Economics*, 119(3):861–898. p8, p13, p143, p152

Campbell, M., Jr., A. J. H., and Hsu, F. (2002). Deep blue. *Artif. Intell.*, 134(1-2):57–83. p12

Cazenave, T., Legras, S., and Ventos, V. (2021). Optimizing $\alpha\mu$. In *2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021*, pages 1–8. IEEE. p12

Cazenave, T. and Ventos, V. (2020). The $\alpha\mu$ search algorithm for the game of bridge. In *Proc. Monte Carlo Search International Workshop*, pages 1–16. Springer. p12, p113

Celli, A. and Gatti, N. (2018). Computational results for extensive-form adversarial team games. In McIlraith, S. A. and Weinberger, K. Q., editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 965–972. AAAI Press. p9, p20, p28, p29

Chatterjee, K., Doyen, L., and Henzinger, T. A. (2013). A survey of partial-observation stochastic parity games. *Formal Methods Syst. Des.*, 43(2):268–284. p11

Chatterjee, K. and Henzinger, T. A. (2012). A survey of stochastic ω-regular games. *J. Comput. Syst. Sci.*, 78(2):394–413. p11

Cloud, A., Wang, A., and Kerr, W. (2023). Anticipatory fictitious play. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 73–81. ijcai.org. p153

Conitzer, V. and Sandholm, T. (2008). New complexity results about nash equilibria. *Games Econ. Behav.*, 63(2):621–641. p10

Conway, J. H. (2000). *On Numbers and Games*. A K Peters/CRC Press, 2nd edition. p9

Coulom, R. (2006). Efficient selectivity and backup operators in monte-carlo tree search. In van den Herik, H. J., Ciancarini, P., and Donkers, H. H. L. M., editors, *Computers and Games, 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers*, volume 4630 of *Lecture Notes in Computer Science*, pages 72–83. Springer. p11

Cowling, P. I., Powley, E. J., and Whitehouse, D. (2012). Information set monte carlo tree search. *IEEE Trans. Comput. Intell. AI Games*, 4(2):120–143. p12

Darwiche, A. and Marquis, P. (2002). A knowledge compilation map. *J. Artif. Intell. Res.*, 17:229–264. p46, p178

Dasgupta, P., Chakrabarti, P. P., and Sarkar, S. C. D. (1996). Searching game trees under a partial order. *Artif. Intell.*, 82(1-2):237–257. p13, p108, p115

Dasgupta, P., Chakrabarti, P. P., and Sarkar, S. C. D. (1999). *Multiobjective heuristic search: an introduction to intelligent search methods for multicriteria optimization*. Computational intelligence. Vieweg. p13

Daskalakis, C., Goldberg, P. W., and Papadimitriou, C. H. (2009). The complexity of computing a nash equilibrium. *SIAM J. Comput.*, 39(1):195–259. p10

Davey, B. A. and Priestley, H. A. (2002). *Introduction to Lattices and Order*. Cambridge University Press, 2nd edition. p165

de Weerd, H., Verbrugge, R., and Verheij, B. (2013). How much does it help to know what she knows you know? an agent-based simulation study. *Artif. Intell.*, 199-200:67–92. p13

Dhami, S. (2019). *The Foundations of Behavioral Economic Analysis Volume IV: Behavioral Game Theory*. Oxford University Press, London, England. p8, p13, p115, p141, p143, p152

Doshi, P., Gmytrasiewicz, P. J., and Durfee, E. H. (2020). Recursively modeling other agents for decision making: A research perspective. *Artif. Intell.*, 279. p13, p152

Doyen, L. and Raskin, J. (2011). Games with imperfect information: theory and algorithms. In Apt, K. R. and Grädel, E., editors, *Lectures in Game Theory for Computer Scientists*, pages 185–212. Cambridge University Press. p111

Edelkamp, S. (2020). Representing and reducing uncertainty for enumerating the belief space to improve endgame play in skat. In *Proc. 24th European Conference on Artificial Intelligence (ECAI 2020)*, pages 395–402. IOS Press. p12, p112

Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, M. Y. (1995). *Reasoning About Knowledge*. MIT Press. p67

Faliszewski, P., Rothe, I., and Rothe, J. (2016). Noncooperative game theory. In Rothe, J., editor, *Economics and Computation, An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division*, Springer texts in business and economics, pages 41–134. Springer. p7, p66

Farina, G., Celli, A., Gatti, N., and Sandholm, T. (2018). Ex ante coordination and collusion in zero-sum multi-player extensive-form games. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9661–9671. p9

Farina, G., Celli, A., Gatti, N., and Sandholm, T. (2021). Connecting optimal ex-ante collusion in teams to extensive-form correlation: Faster algorithms and positive complexity results. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 3164–3173. PMLR. p9

Frank, I. and Basin, D. A. (1998). Search in games with incomplete information: A case study using bridge card play. *Artif. Intell.*, 100(1-2):87–123. p12, p90, p95, p96, p122, p139

Frank, I. and Basin, D. A. (2001). A theoretical and empirical investigation of search in imperfect information games. *Theor. Comput. Sci.*, 252(1-2):217–256. p12, p41, p76, p77, p93

Fédération Française de Bridge (2013). *Le bridge français: Tome 1*. Éditions Pole. p167

Geffner, H. and Bonet, B. (2013). *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. p6

Genesereth, M. R. and Thielscher, M. (2014). *General Game Playing*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. p48

Ghallab, M., Nau, D. S., and Traverso, P. (2004). *Automated planning - theory and practice*. Elsevier. p6

Ghallab, M., Nau, D. S., and Traverso, P. (2016). *Automated Planning and Acting*. Cambridge University Press. p6

Gilboa, I. and Zemel, E. (1989). Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1(1):80–93. p10

Gimbert, H., Paul, S., and Srivathsan, B. (2020). A bridge between polynomial optimization and games with imperfect recall. In Seghrouchni, A. E. F., Sukthankar, G., An, B., and Yorke-Smith, N., editors, *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, pages 456–464. International Foundation for Autonomous Agents and Multiagent Systems. p11, p33, p64

Ginsberg, M. and Jaffray, A. (2002). Alpha-beta pruning under partial orders. In Nowakowski, R., editor, *More Games of No Chance*, number 42 in Mathematical Sciences Research Institute Publications, pages 37–48. Cambridge University Press. p112, p113, p115, p116, p117

Ginsberg, M. L. (2001). GIB: imperfect information in a computationally challenging game. *J. Artif. Intell. Res.*, 14:303–358. p12, p93, p97, p109, p113, p114

Gmytrasiewicz, P. J. and Doshi, P. (2005). A framework for sequential planning in multi-agent settings. *J. Artif. Intell. Res.*, 24:49–79. p13

Goldsmith, J. and Mundhenk, M. (2007). Competition adds complexity. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 561–568. Curran Associates, Inc. p11, p41, p51

Grädel, E. (1990). Domino games and complexity. *SIAM J. Comput.*, 19(5):787–804. p41

Halpern, J. Y. (2011). Beyond nash equilibrium: solution concepts for the 21st century. In Apt, K. R. and Grädel, E., editors, *Lectures in Game Theory for Computer Scientists*, pages 264–290. Cambridge University Press. p9

Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In McGuinness, D. L. and Ferguson, G., editors, *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 709–715. AAAI Press / The MIT Press. p6

Haworth, G. and Hernandez, N. (2021). The 20th top chess engine championship, TCEC20. *ICGA Journal*, 43(1):62–73. p12, p113

Hearn, R. A. and Demaine, E. D. (2009). *Games, Puzzles and Computation*. A K Peters. p11

Huth, M. and Ryan, M. D. (2004). *Logic in computer science: modelling and reasoning about systems (2. ed.)*. Cambridge University Press. p67

Iida, H., Uiterwijk, J. W. H. M., van den Herik, H. J., and Herschberg, I. S. (1993). Potential applications of opponent-model search, part 1: The domain of applicability. *J. Int. Comput. Games Assoc.*, 16(4):201–208. p13, p144, p145, p147, p152

Iida, H., Uiterwijk, J. W. H. M., van den Herik, H. J., and Herschberg, I. S. (1994). Potential applications of opponent-model search, part 2: Risks and strategies. *J. Int. Comput. Games Assoc.*, 17(1):10–14. p13, p144

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134. p6

Karpin, F. L. (1977). *Psychological Strategy in Contract Bridge: The Techniques of Deception and Harassment in Bidding and Play*. Dover Publications. p169

Khemani, D., Singh, S., and AC, I. (2018). Contract bridge: Multi-agent adversarial planning in an uncertain environment. In *Poster Collection of the Sixth Annual Conference on Advances in Cognitive Systems. ACS (Online available at www. cogsys. org/papers/ACSvol6/posters/Khemani. pdf)*. p12

Kissmann, P. and Edelkamp, S. (2009). Solving fully-observable non-deterministic planning problems via translation into a general game. In Mertsching, B., Hund, M., and Zaheer Aziz, M., editors, *Proc. 32nd Annual German Conference on Artificial Intelligence (KI 2009)*, pages 1–8. Springer. p112

Knuth, D. E. (2011). *Art of Computer Programming, Volume 4A, Combinatorial Algorithms, Part 1*. Addison-Wesley. p164

Knuth, D. E. and Moore, R. W. (1975). An analysis of alpha-beta pruning. *Artif. Intell.*, 6(4):293–326. p11, p39, p113, p116, p117

Kochenderfer, M. J. (2015). *Decision Making Under Uncertainty: Theory and Application*. MIT Press. p5

Kochenderfer, M. J., Wheeler, T. A., and Wray, K. H. (2022). *Algorithms for Decision Making*. The MIT Press. p5, p7

Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M., editors, *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer. p11

Koller, D. and Megiddo, N. (1992). The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior*, 4(4):528–552. p10, p11, p26, p33, p36, p37, p39, p40, p41, p55, p56, p63, p64, p145, p146, p149, p150

Koller, D., Megiddo, N., and von Stengel, B. (1996). Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14(2):247–259. p10, p145

Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artif. Intell.*, 27(1):97–109. p11

Kovarík, V., Schmid, M., Burch, N., Bowling, M., and Lisý, V. (2022). Rethinking formal models of partially observable multiagent decision making. *Artif. Intell.*, 303:103645. p10

Kozen, D. (2006). *Theory of Computation*. Texts in Computer Science. Springer. p10

Kuhn, H. W. (1953). Extensive games and the problem of information. In *Contributions to the Theory of Games (AM-28), Volume II*, pages 193–216. Princeton University Press, Princeton. p27, p28

Kupferschmid, S. and Helmert, M. (2006). A skat player based on monte-carlo simulation. In *Proc. 5th International Conference on Computers and Games (CG 2006)*, pages 135–147. Springer. p12, p112

Larsson, U., editor (2019). *Games of No Chance 5*, volume 70 of *Mathematical Sciences Research Institute Publications*. Cambridge University Press. p10

Levy, D. N. (1989). The million pound bridge program. *Heuristic Programming in Artificial Intelligence: The First Computer Olympiad*, pages 95–103. p12, p112

Li, J., Zanuttini, B., Cazenave, T., and Ventos, V. (2022). Generalisation of alpha-beta search for AND-OR graphs with partially ordered values. In Raedt, L. D., editor, *Proc. Thirty-First International Joint Conference on Artificial Intelligence (IJCAI 2022)*, pages 4769–4775. ijcai.org. p94, p112

Li, J., Zanuttini, B., and Ventos, V. (2024). Opponent-model search in games with incomplete information. In Dy, J. and Natarajan, S., editors, *Proc. Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)*. AAAI Press. To appear. p144

Loddo, J. and Saiu, L. (2010). How to correctly prune tropical trees. In Autexier, S., Calmet, J., Delahaye, D., Ion, P. D. F., Rideau, L., Rioboo, R., and Sexton, A. P., editors, *Proc. 10th International Conference on Intelligent Computer Mathematics*, volume 6167 of *Lecture Notes in Computer Science*, pages 101–115. Springer. p115

Long, J. R., Sturtevant, N. R., Buro, M., and Furtak, T. (2010). Understanding the success of perfect information monte carlo sampling in game tree search. In Fox, M. and Poole, D., editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, pages 134–140. AAAI Press. p12

Madani, O., Hanks, S., and Condon, A. (2003). On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif. Intell.*, 147(1-2):5–34. p11

Marsland, T. A. (1986). A review of game-tree pruning. *J. Int. Comput. Games Assoc.*, 9(1):3–19. p11, p112, p113, p114, p116, p117, p119

Maschler, M., Solan, E., and Zamir, S. (2020). *Game Theory*. Cambridge University Press, 2nd edition. p7, p8, p9, p12, p15, p21, p25, p27, p66, p67, p68

McMahan, H. B., Gordon, G. J., and Blum, A. (2003). Planning in the presence of cost functions controlled by an adversary. In Fawcett, T. and Mishra, N., editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 536–543. AAAI Press. p10, p53, p148, p152, p153

Moore, C. and Mertens, S. (2011). *The Nature of Computation*. Oxford University Press, London, England. p10

Moulin, H. (1985). Choice functions over a finite set: a summary. *Social Choice and Welfare*, 2(2):147–160. p102, p103, p104

Mundhenk, M., Goldsmith, J., Lusena, C., and Allender, E. (2000). Complexity of finite-horizon markov decision process problems. *J. ACM*, 47(4):681–720. p11

Nagel, R. (1995). Unraveling in guessing games: An experimental study. *The American Economic Review*, 85(5):1313–1326. p8

Nash, J. (1951). Non-cooperative games. *Annals of Mathematics*, 54(2):286–295. p8

Nash, J. F. (1950). Equilibrium points in ¡i¿n¡/i¿-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49. p8

Nashed, S. B. and Zilberstein, S. (2022). A survey of opponent modeling in adversarial domains. *J. Artif. Intell. Res.*, 73:277–327. p13

Nasu, Y. (2018). Efficiently updatable neural-network-based evaluation functions for computer shogi. *The 28th World Computer Shogi Championship Appeal Document*. p113

Nisan, N., Roughgarden, T., Tardos, É., and Vazirani, V. V., editors (2007). *Algorithmic Game Theory*. Cambridge University Press. p7

Niveau, A. and Zanuttini, B. (2016). Efficient representations for the modal logic S5. In Kambhampati, S., editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1223–1229. IJCAI/AAAI Press. p138

Nowakowski, R. J., editor (1996). *Games of No Chance*, volume 29 of *Mathematical Sciences Research Institute Publications*. Cambridge University Press. p10

Nowakowski, R. J., editor (2002). *More Games of No Chance*, volume 42 of *Mathematical Sciences Research Institute Publications*. Cambridge University Press. p10

Nowakowski, R. J., editor (2015). *Games of No Chance 4*, volume 63 of *Mathematical Sciences Research Institute Publications*. Cambridge University Press. p10

Oliehoek, F. A. and Amato, C. (2016). *A Concise Introduction to Decentralized POMDPs*. Springer Briefs in Intelligent Systems. Springer. p6

Palacios, H. and Geffner, H. (2009). Compiling uncertainty away in conformant planning problems with bounded width. *J. Artif. Intell. Res.*, 35:623–675. p73

Papadimitriou, C. H. (1994). *Computational complexity*. Addison-Wesley. p164

Pearce, D. G. (1984). Rationalizable strategic behavior and the problem of perfection. *Econometrica*, 52(4):1029–1050. p8

Pearl, J. (1982). The solution for the branching factor of the alpha-beta pruning algorithm and its optimality. *Commun. ACM*, 25(8):559–564. p11, p113

Perea, A. (2010). Backward induction versus forward induction reasoning. *Games*, 1(3):168–188. p8

Perea, A. (2012). *Epistemic Game Theory: Reasoning and Choice*. Cambridge University Press. p7, p8, p13, p142, p143

Perea, A. (2024). *From Decision Theory to Game Theory: Reasoning about Decisions of Others*. Cambridge University Press. To be published. p7

Perifel, S. (2014). *Complexité algorithmique*. Ellipses. p10

Pérolat, J., Vylder, B. D., Hennes, D., Tarassov, E., Strub, F., de Boer, V., Muller, P., Connor, J. T., Burch, N., Anthony, T. W., McAleer, S., Elie, R., Cen, S. H., Wang, Z., Gruslys, A., Malysheva, A., Khan, M., Ozair, S., Timbers, F., Pohlen, T., Eccles, T., Rowland, M., Lanctot, M., Lespiau, J., Piot, B., Omidshafiei, S., Lockhart, E., Sifre, L., Beauguerlange, N., Munos, R., Silver, D., Singh, S., Hassabis, D., and Tuyls, K. (2022). Mastering the game of stratego with model-free multiagent reinforcement learning. *CoRR*, abs/2206.15378. p12

Peterson, G., Reif, J., and Azhar, S. (2001). Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, 41(7):957–992. p11, p34, p45, p50

Piccione, M. and Rubinstein, A. (1997). On the interpretation of decision problems with imperfect recall. *Games and Economic Behavior*, 20(1):3–24. p33

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley. p6

Rebstock, D., Solinas, C., Buro, M., and Sturtevant, N. R. (2019). Policy based inference in trick-taking card games. In *IEEE Conference on Games, CoG 2019, London, United Kingdom, August 20-23, 2019*, pages 1–8. IEEE. p12, p13, p112

Rintanen, J. (2004). Complexity of planning with partial observability. In Zilberstein, S., Koehler, J., and Koenig, S., editors, *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), June 3-7 2004, Whistler, British Columbia, Canada*, pages 345–354. AAAI. p6, p11, p108

Roijers, D. M. and Whiteson, S. (2017). *Multi-Objective Decision Making*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. p13

Roughgarden, T. and Iwama, K. (2017). Twenty lectures on algorithmic game theory. *Bull. EATCS*, 122. p7

Rousset, M.-C. (2022). Nook, robot de bridge. `https://www.lemonde.fr/blog/binaire/2022/06/28/nook-robot-de-bridge/`. Accessed: 2023-10-23. p12

Russell, S. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition. p5, p11

Schaefer, M. and Umans, C. (2002). Completeness in the polynomial-time hierarchy: A compendium. *SIGACT news*, 33(3):32–49. p42

Schaeffer, J. (2008). *One Jump Ahead: Computer Perfection at Checkers*. Springer New York. p12

Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P., and Sutphen, S. (2007). Checkers is solved. *Science*, 317(5844):1518–1522. p12, p112

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T. P., and Silver, D. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nat.*, 588(7839):604–609. p12

Schwarzentruber, F. (2019). The complexity of tiling problems. *CoRR*, abs/1907.00102. p42

Seuken, S. and Zilberstein, S. (2008). Formal models and algorithms for decentralized decision making under uncertainty. *Auton. Agents Multi Agent Syst.*, 17(2):190–250. p6

Shannon, C. E. (1949). Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28(4):656–715. p78

Shoham, Y. and Leyton-Brown, K. (2009). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press. p7, p8

Siegel, A. N. (2013). *Combinatorial game theory*. American Mathematical Society. p10

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489. p12

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T. P., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nat.*, 550(7676):354–359. p12

Sipser, M. (2012). *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition. p10

Stahl, D. O. and Wilson, P. W. (1995). On players' models of other players: Theory and experimental evidence. *Games and Economic Behavior*, 10(1):218–254. p8, p152

Stockman, G. C. (1979). A minimax algorithm better than alpha-beta? *Artif. Intell.*, 12(2):179–196. p11

Stockmeyer, L. J. and Meyer, A. R. (1973). Word problems requiring exponential time: Preliminary report. In Aho, A. V., Borodin, A., Constable, R. L., Floyd, R. W., Harrison, M. A., Karp, R. M., and Strong, H. R., editors, *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA*, pages 1–9. ACM. p50

Sturtevant, N. R. and Bowling, M. H. (2006). Robust game play against unknown opponents. In Nakashima, H., Wellman, M. P., Weiss, G., and Stone, P., editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 713–719. ACM. p144

Sturtevant, N. R. and White, A. M. (2006). Feature construction for reinforcement learning in hearts. In *Proc. 5th International Conference on Computers and Games (CG 2006)*, pages 122–134. Springer. p12, p113

Sturtevant, N. R., Zinkevich, M., and Bowling, M. H. (2006). Prob-maxn: Playing n-player games with opponent models. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 1057–1063. AAAI Press. p13, p144, p153

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning*. Adaptive Computation and Machine Learning series. MIT Press, Cambridge, MA, 2nd edition. p6, p11

Swiechowski, M., Godlewski, K., Sawicki, B., and Mandziuk, J. (2023). Monte carlo tree search: a review of recent modifications and applications. *Artif. Intell. Rev.*, 56(3):2497–2562. p11

Tewolde, E., Oesterheld, C., Conitzer, V., and Goldberg, P. W. (2023). The computational complexity of single-player imperfect-recall games. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 2878–2887. ijcai.org. p33

Tseitin, G. S. (1983). On the complexity of derivation in propositional calculus. In Siekmann, J. and Wrightson, G., editors, *Automation of reasoning: 2: Classical papers on computational logic 1967–1970*, pages 466–483. Springer. p177

Umans, C. (1999). Hardness of approximating Σ_2^p minimization problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 465–474. IEEE Computer Society. p81

van Damme, E. (1991). *Stability and Perfection of Nash Equilibria*. Springer Berlin Heidelberg. p8

van den Herik, H., Uiterwijk, J. W., and van Rijswijck, J. (2002). Games solved: Now and in the future. *Artif. Intell.*, 134(1-2):277–311. p112

van Ditmarsch, H., Halpern, J. Y., van der Hoek, W., and Kooi, B., editors (2015). *Handbook of epistemic logic*. College Publications. p67

van Emde Boas, P. (1997). The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, pages 331–363. CRC Press. p42

von Neumann, J. and Morgenstern, O. (1994). *Theory of Games and Economic Behavior*. Princeton University Press. p7

von Stengel, B. (1996). Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2):220–246. p10, p145

von Stengel, B. and Koller, D. (1997). Team-maxmin equilibria. *Games and Economic Behavior*, 21(1):309–321. p9, p28

Wright, J. R. and Leyton-Brown, K. (2019). Level-0 models for predicting human behavior in games. *J. Artif. Intell. Res.*, 64:357–383. p13, p152

You, Y., Thomas, V., Colas, F., Alami, R., and Buffet, O. (2023). Robust robot planning for human-robot collaboration. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 9793–9799. IEEE. p13

Zermelo, E. (1913). 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. p11

Zhang, B. H., Farina, G., and Sandholm, T. (2023). Team belief DAG: generalizing the sequence form to team games for fast computation of correlated team max-min equilibria via regret minimization. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 40996–41018. PMLR. p9, p11, p63, p64

Zhang, B. H. and Sandholm, T. (2022). Team correlated equilibria in zero-sum extensive-form games via tree decompositions. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 5252–5259. AAAI Press. p9

Zhang, Y., An, B., and Cerný, J. (2021). Computing ex ante coordinated team-maxmin equilibria in zero-sum multiplayer extensive-form games. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 5813–5821. AAAI Press. p9

Zinkevich, M., Johanson, M., Bowling, M. H., and Piccione, C. (2007). Regret minimization in games with incomplete information. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1729–1736. Curran Associates, Inc. p12

Abstracts

Jeux à information incomplète : complexité, algorithmique, raisonnement

Dans cette thèse, on étudie les jeux à information incomplète. On commence par établir un paysage complet de la complexité du calcul des stratégies pures optimales pour différentes sous-catégories de jeux, lorsque les jeux sont explicitement donnés en entrée. On étudie ensuite la complexité lorsque les jeux sont représentés de façon compacte (par leurs règles de jeu, par exemple). Pour ce faire, on conçoit deux formalismes pour ces représentations compactes. Dans la suite, on se consacre aux jeux à information incomplète, en proposant d'abord un nouveau formalisme, nommé jeu combinatoire à information incomplète, qui regroupe les jeux sans hasard (sauf un tirage aléatoire initial) et avec uniquement des actions publiques. Ce nouveau formalisme capture la notion de l'information et de la connaissance des joueurs mieux que la forme extensive. Puis, on étudie des algorithmes et leurs optimisations pour résoudre les jeux combinatoires à information incomplète ; certains algorithmes que l'on propose sont applicables au-delà de ces jeux. Dans la dernière partie, on présente un travail en cours, qui consiste à modéliser les raisonnements récursifs et différents types de connaissances sur le comportement des adversaires dans les jeux à information incomplète.

Mots-clefs : jeu sous forme extensive, jeu à information incomplète, maxmin, recherche arborescente, modèle d'opposant

Games with incomplete information: complexity, algorithmics, reasoning

In this dissertation, we study games with incomplete information. We begin by establishing a complete landscape of the complexity of computing optimal pure strategies for different subclasses of games, when games are given explicitly as input. We then study the complexity when games are represented compactly (e.g. by their game rules). For this, we design two formalisms for such compact representations. Then we concentrate on games with incomplete information, by first proposing a new formalism called combinatorial game with incomplete information, which encompasses games of no chance (apart from a random initial drawing) and with only public actions. This new formalism captures the notion of information and knowledge of the players in a game better than the extensive form. Next, we study algorithms and their optimisations for solving combinatorial games with incomplete information; some of these algorithms are applicable beyond these games. In the last part, we present a work in progress that concerns the modelling of recursive reasoning and different types of knowledge about the behaviour of the opponents in games with incomplete information.

Keywords: extensive-form game, game with incomplete information, maxmin, game tree search, opponent model