POLITECNICO DI MILANO
Facoltà di Ingegneria dell'Informazione
Dipartimento di Elettronica e Informazione

Strong Nash Equilibrium: Smoothed Computational Complexity and Algorithms for Games with Three or More Players

Relatore: Prof. Nicola GATTI

Tesi di Laurea di:
Gianluca STAFFIERO
Matr. n. 749867

Anno Accademico 2012-2013

A chi mi ha sostenuto
nei momenti di difficoltà

Abstract

The most important solution concept in game theory is the *Nash equilibrium* (NE). However, this solution concept fails when agents can form coalition, even in the case of two–agent games. *Strong Nash equilibrium* (SNE) is an appealing solution concept that refines NE to capture such a case. An SNE must simultaneously be a NE and the optimal solution of multiple non-convex optimization problems. This makes even the derivation of necessary and sufficient equilibrium constraints in a mathematical programming fashion difficult. Moreover it is known that finding an SNE is a $\mathcal{NP}$–complete problem. In the first part of this work we study the properties of mixed–strategy SNEs in bimatrix game, providing two main contributions. We show that if there exists an SNE in a bimatrix game, then the payoffs of the actions in the SNE support must lay on the same line in agents' utilities space. Furthermore, we show that finding an SNE is a problem in smoothed–$\mathcal{P}$, admitting a deterministic support-enumeration algorithm with smoothed polynomial running time. In the second part of the work we derive different sets of conditions for a strategy to be an SNE. We show that forcing an SNE to be resilient only to pure strategy deviations by coalitions, unlike for NEs, is only a necessary condition. We also show that the application of Karush–Kuhn–Tucker (KKT) conditions leads to another set of necessary conditions that are not sufficient. Finally we can show that forcing the Pareto efficiency of a NE for each coalition with respect to coalition correlated–strategies is sufficient but not necessary. In the end we develop a tree search algorithm for SNE finding which, at each node, calls an oracle to suggest a candidate SNE and then verifies the candidate. We show that our necessary conditions can be leveraged to make the oracle more powerful. Experiments show that the new conditions reduce search tree size compared to using NE conditions alone, however it is necessary to use an incomplete nonlinear solver, which affects the soundness and the completeness of the whole algorithm.

Keywords: Strong Nash equilibrium, Smoothed complexity, Non–linear programming, Game theory, Algorithms

Sommario

La ricerca degli *equilibri di Nash* (NE) è un problema di grande importanza nel campo dell'intelligenza artificiale. Tuttavia, il concetto di NE è valido solo quando gli agenti non possono formare coalizioni. L'*equilibrio di Nash forte* (SNE) estende il concetto di NE in modo da catturare proprio queste situazioni. Uno SNE deve essere contemporaneamente un NE e la soluzione di diversi problemi non convessi di ottimizzazione. Questo rende difficile derivare un insieme finito di vincoli necessari e sufficienti affinchè una strategia sia uno SNE. Inoltre è noto che trovare uno SNE è un problema $\mathcal{NP}$–completo. Nella prima parte di questo testo vengono studiate le proprietà degli SNE e vengono forniti due contributi principali. Dapprima viene mostrato che se esiste uno SNE in un gioco a due giocatori allora i payoff delle azioni nel supporto dell'equilibrio devono essere allineati nello spazio dell'utilità degli agenti. In secondo luogo viene mostrato che trovare uno SNE in un gioco a due giocatori è un problema che ha classe di complessità smoothed–$\mathcal{P}$. Nella seconda parte di questa tesi vengono derivati diversi insiemi di condizioni affinché una strategia sia uno SNE. Si dimostra che è possibile ricavare un insieme di condizioni necessarie, ma non sufficienti, o imponendo ad una strategia di essere robusta rispetto a deviazioni multilaterali in strategie pure, o utilizzando le condizioni di Karush-Kuhn-Tucker. Inoltre si dimostra che imporre la Pareto efficienza di un NE in strategie correlate, per ogni possibile coalizione, è una condizione sufficiente ma non necessaria. Per concludere viene sviluppato un algoritmo branch–and–bound per la ricerca di uno SNE che, ad ogni iterazione, utilizza un oracolo per generare un candidato e verifica se tale candidato è uno SNE. Viene mostrato che utilizzando gli insiemi di condizioni necessarie individuati si ottengono degli oracoli più performanti, infatti si riduce in maniera significativa il numero di branch rispetto all'uso di oracoli che genereano candidati che sono NE. Tuttavia si rende necessario l'utilizzo di un risolutore non lineare e non completo, il che ha un impatto sulla completezza e la correttezza dell'algorimo.

Parole chiave: equilibrio di Nash forte, Complessità smoothed, Programmazione non lineare, Teoria dei giochi, Algoritmi

Contents

Abstract **I**

Sommario **III**

1 Introduction **1**

2 Prior Art **5**
2.1 Game Theoretical Preliminaries 5
2.1.1 Self-interested agents and utility function 6
2.1.2 Strategic form game 6
2.1.3 Strategies 7
2.1.4 Expected Utility 7
2.1.5 Correlated strategies 8
2.1.6 Bimatrix games 9
2.2 Solution Concepts 10
2.2.1 Pareto efficiency 10
2.2.2 Nash equilibrium 12
2.2.3 Finding Nash equilibria 13
2.2.4 Strong Nash equilibrium 14
2.2.5 Finding strong Nash equilibria 16
2.3 Nonlinear Programming 16
2.3.1 Karush–Kuhn–Tucker Conditions 17

3 Properties of strong Nash equilibria **19**
3.1 Pareto optimality in bimatrix games 20
3.1.1 Necessary conditions 20
3.1.2 Maximal support strategies 23
3.2 Mixed strategy Nash equilibrium conditions 23
3.3 Strong Nash Necessary Conditions 25
3.4 Games with mixed strategy strong Nash equilibrium 26
3.4.1 Bimatrix games with two actions per agent 26

3.4.2 Generic bimatrix games 32
3.5 Smoothed complexity 34

4 Mathematical equilibrium constraints 37
4.1 Strong Nash equilibrium conditions 38
4.2 Resilience to pure multilateral deviations 40
4.3 Karush–Kuhn–Tucker Conditions 43
4.4 Correlated Pareto frontier 46
4.5 Relationships between the solutions 49

5 Iterative Algorithm for Strong Nash equilibrium 53
5.1 Iterative strong Nash algorithm 53
5.1.1 Operative principles 54
5.1.2 Algorithm details 55
5.1.3 Implementation 56
5.2 Nonlinear solver issues 56
5.3 Experimental results 58
5.3.1 Test settings and objectives 58
5.3.2 Test results 59

6 Conclusions and Future Works 63

Bibliografia 66

A AMPL Models for Three–Agents Games 69

List of Figures

2.1 The Pareto frontier of the game described in Table 2.2. 16

3.1 Examples used in the proof of Theorem 3.4.1. 28
3.2 The Pareto frontier of the game described in Table 3.2. 30
3.3 The Pareto frontier of the game described in Table 3.3. 31
3.4 The Pareto frontier of the game described in Table 3.4. 33

4.1 The Pareto frontier of the game described in Table 4.1. 42
4.2 The Pareto frontier of the game described in Table 4.2. 45
4.3 The Pareto frontier of the game described in Table 4.3. 49
4.4 The Pareto frontier of the game described in Table 4.4. 50
4.5 The Pareto frontier of the game described in Table 4.5. 51
4.6 The Hasse diagram representing the partial order induced by the (strict) inclusion relation among the sets $corrSNE$, SNE, $NEKKT \cap NEPMD$, $NEKKT$, $NEPMD$ and NE. 52

List of Tables

2.1 A generic bimatrix game where there are three actions available for the first agent and two for the second. 8
2.2 Utility matrix of Prisoner's dilemma 15

3.1 A generic bimatrix game with two actions per agent. 27
3.2 Example of two–agent game which satisfies Conditions 3.31–3.30 but does not have a mixed–strategy strong Nash equilibrium. 29
3.3 Example of two–agent game with a mixed–strategy strong Nash equilibrium. 31
3.4 Example of two–agent game, with more than two actions per agent, with a mixed–strategy strong Nash equilibrium. 33

4.1 Example of a game with a Nash equilibrium which is resilient to pure–strategies multilateral deviations but is not a strong Nash equilibrium, being Pareto dominated. 42
4.2 Example of a game with a Nash equilibrium that satisfies Karush–Kuhn–Tucker conditions even though it is Pareto dominated. 45
4.3 Example of a game with a Nash equilibrium which does not lie on the correlated–strategy Pareto frontier. 48
4.4 Example of a game with a Nash equilibrium which is resilient to pure–strategies multilateral deviations but does not satisfy the Karush–Kuhn–Tucker conditions. 49
4.5 Example of a game with a Nash equilibrium which satisfies the Karush–Kuhn–Tucker conditions but is not resilient to pure–strategies multilateral deviations. 50

5.1 Experimental results: properties of Nash equilibrium–finding with SNOPT 60

5.2 Experimental results: accuracy of SNE–finding algorithm, given a strong Nash equilibrium exists. 61
5.3 Experimental results: percentage of errors of SNE–finding algorithm, given a strong Nash equilibrium does not exists. . 61

Chapter 1

Introduction

The computational characterization of game–theoretic solution concepts is a central topic in artificial intelligence with the aim of developing computationally efficient tools to deal with strategic interaction situations among rational agents. The most important solution concept provided by game theory is *Nash equilibrium*. Although every finite game admits at least a Nash equilibrium in mixed strategies, game theory itself does not provide any tool to find such equilibria. Computer science provides a number of algorithms and characterizes the complexity of such a problem. Many papers have focused on the computational study of *Nash equilibrium* [26], showing that searching for it is $\mathcal{PPAD}$–complete [8] even with two agents [6] and designing various algorithms. Two–agent games can be solved by linear complementarity programming [17], support enumeration [22], or mixed–integer linear programming [24]. With more agents, classical methods based on a nonlinear complementarity programing, simplicial subdivision, or homotopy can be used [26], as can support enumeration [28].

Even though $\mathcal{PPAD} \subseteq \mathcal{NP}$, it is generally believed that $\mathcal{PPAD} \neq \mathcal{P}$ and therefore that the worst–case complexity of finding a Nash equilibrium is exponential in the size of the game. Furthemore, bimatrix games do not have a fully polynomial–time approximation scheme unless $\mathcal{PPAD} \subseteq \mathcal{P}$ [6] and finding a Nash equilibrium in two–agent games is not in smoothed–$\mathcal{P}$ unless $\mathcal{PPAD} \subseteq \mathcal{RP}$ [5] and, therefore, by definition of smoothed complexity, game instances keep to be hard even if subjected to small perturbations. For two–agent games, instances [25] that require exponential time when solved with a number of algorithms [17, 22] are known. However, these instances are unstable, becoming easy when small perturbations are applied [11].

Nash equilibrium captures the situation in which no agent can gain more by unilaterally changing her strategy, while, when agents can form coalitions

and change their strategy multilaterally in a coordinate way, *strong Nash equilibrium* concept is adopted [1]. It captures the situation in which agents can form coalitions and change their strategies multilaterally in a coordinated way. Strong Nash equilibrium presents different properties from Nash equilibrium: a strong Nash equilibrium is not assured to exist and searching for it is $\mathcal{NP}$–complete: the hardness is shown in [7], while the membership to $\mathcal{NP}$ is shown in [12]. Few computational results are known about the strong Nash equilibria. An strong Nash equilibrium must be simultaneously an Nash equilibrium and the optimal solution of multiple non–convex optimization problems [15]. This makes even the derivation of necessary and sufficient equilibrium constraints, in mathematical programming fashion, a difficult and currently open task. The literature provides a number of algorithm for finding pure–strategy equilibria only for specific classes of games, e.g., congestion games [16, 14, 23, 15], connection games [9], and maxcut games [13]. However, hardness is due to the existence of mixed–strategy strong Nash equilibria, given that the pure–strategy ones can be found in polynomial time by combining support enumeration and verification. The only algorithm working also with mixed strategies is provided in [12], however it is limited to the class of two–agents games.

In the first part of this thesis we provide the following main contributions.

- We show that, if there is a mixed–strategy strong Nash equilibrium, then the payoffs restricted to the actions in the support must satisfy a restrictive property: they must lay on the same line in agents' utilities space.

- We show that finding a strong Nash equilibrium is in smoothed–$\mathcal{P}$ with two agents, admitting a deterministic support–enumeration algorithm with smoothed polynomial running time, and therefore hard instances are isolated.

It is interesting to notice that the last point implies that, except for a space of the parameters with null measure, games admit only pure–strategy strong Nash equilibria.

In the second part of this work we focus on deriving a set of conditions for a strategy to be a strong Nash equilibrium and we design a general algorithm for strong Nash equilibrium finding. Remark that the only prior algorithm that works also for mixed strategies is only for two–agent games [12]. It is a special kind of tree search algorithm, and at each node it calls an $\mathcal{NP}$–complete oracle (a variation of *MIP Nash* [24]) that returns an Nash equilibrium, if one exists, in a given subspace of the agents' utilities and

then verifies whether the returned equilibrium is a strong Nash equilibrium. In this work we provide the following contributions.

- We provide a mixed–integer nonlinear program to find Nash equilibria that are resilient to pure strategy multilateral deviations. We prove that, unlike in the case of Nash equilibrium, it is only necessary (not sufficient) for a strategy to be a strong Nash equilibrium.
- We provide a nonlinear program to find an NE that satisfies Karush–Kuhn–Tucker conditions [19]. We prove that it is necessary, but not sufficient, for a strategy to be a strong Nash equilibrium.
- We provide a nonlinear program in which a strategy profile is forced to be Pareto efficient with respect to coalition correlated–strategies. We prove that this is sufficient, but not necessary, for a strategy to be a strong Nash equilibrium.
- We characterize the relationships between the solutions of these three formulations, the set of Nash equilibria, and the set of strong Nash equilibria.
- We extend the prior strong Nash equilibrium–finding tree search algorithm [12] to multiple agents. We do this by introducing a generalization to the tree search framework and by leveraging our necessary conditions in the oracle that is used at each search node.
- We show how the use of a nonlinear incomplete solver affects the soundness and the completeness of our algorithm.
- We conduct experiments that validate the overall tree search approach and shows the benefits of our new nonlinear necessary conditions in the oracle.

Chapter 2

Prior Art

This chapter is composed of three main parts. In the first section some basic game theoretical concepts are introduced. These preliminaries are widely used through the rest of the work. In the second part we introduce three main solution concepts: the *Pareto optimality*, the *Nash equilibrium* and the *strong Nash equilibrium*. Furthermore we briefly discuss the computational complexity of the general problem of finding these solutions and some of the most known algorithms. In the last section we introduce some basics on the nonlinear programming with a particular focus on the Karush–Kuhn–Tucker conditions, that is an approach to nonlinear programming that generalizes the method of Lagrange multipliers. This technique is interesting as it allow us to we write a set of necessary conditions for Pareto optimality.

2.1 Game Theoretical Preliminaries

Game theory is the mathematical study of interaction among independent, self–interested agents [26]. It has been applied to disciplines as diverse as economics (historically, its main area of application), political science, biology, psychology, linguistics and computer science. The *noncooperative game theory* has become the dominant branch of game theory, and the *normal-form games* are a canonical representation in this discipline. In noncooperative game theory the basic modeling unit is the individual, including his beliefs, preferences, and possible actions. In this section we are providing the basic definitions and properties which will be used through the rest of the work.

2.1.1 Self-interested agents and utility function

The notion of self-interested agent means that each agent has his own description of which states of the world he prefers and that he acts in an attempt to bring about these states of the world. The dominant approach to model an agent's interests is utility theory. This theoretical approach aims to quantify an agent's degree of preference across a set of available alternatives.

Definition 2.1.1. A *utility function* is a mapping from states of the world to real numbers [26].

These numbers are interpreted as measures of an agent's level of satisfaction in the given states. When the agent is uncertain about which state of the world he faces, his utility is defined as the expected value of his utility function with respect to the appropriate probability distribution over states.

2.1.2 Strategic form game

The strategic form, also known as normal or matrix form, is the most familiar representation of strategic interactions in game theory. A game written in this way amounts to a representation of every player's utility for every state of the world, in the special case where states of the world depend only on the players' combined actions. In general most other representations of interest can be reduced to a strategic form game, therefore the normal-form representation is the most fundamental in game theory.

Definition 2.1.2. A *strategic–form game* is a $G = (N, A, U)$ where: [26]

- $N = \{1, \ldots, n\}$ is the set of agents (we denote by i a generic agent),
- $A = A_1 \times \cdots \times A_n$ is the set of agents' actions where A_i is a finite set of actions available for agent i; moreover we denote a generic action by a, and by m_i the number of actions in A_i,
- $U = \{U_1 \ldots, U_n\}$ is the set of agents' utility arrays where $U_i(a_1, \ldots, a_n)$ is agent i's utility when the agents play actions $a_1, \ldots, a_n$.

A natural way to represent games is via an n–dimensional matrix.

It is easy to prove that for every couple of strategic form game the following lemma holds:

Lemma 2.1.3. *Two strategic games are equivalent if the utility matrix of one can be obtained by applying the same affine transformation to every entry of the utility matrix of the other.*

Hence, each strategic form game has an equivalent game where the following equations hold:

$$\max_{a_1,\ldots,a_n} \{U_i(a_1,\ldots,a_n)\} = 1, \forall i \in N$$

$$\min_{a_1,\ldots,a_n} \{U_i(a_1,\ldots,a_n)\} = 0, \forall i \in N$$

2.1.3 Strategies

The first kind of strategy is to select a single action and play it. We call such a strategy a *pure strategy*, notice that each action $a \in A_i$ is said to be a pure strategy for agent i. We call a choice of pure strategy for each agent a *pure–strategy profile* or pure joint strategy. Players can also randomize over the set of available actions according to some probability distribution. Such a strategy is called a *mixed strategy*. We denote by $x_i(a_j)$, or x_{i,a_j} for a more compact representation, the probability with which agent i plays action $a_j \in A_i$ and by $\mathbf{x}_i$ the vector of probabilities $x_i(a_j)$ of agent i. Then we define a mixed strategy for a normal-form game as follows.

Definition 2.1.4. Let (N, A, U) be a strategic–form game, a *mixed strategy* $\mathbf{x}_i$ for agent i is a probability distribution over the agent's set of actions A_i, where $x_{i,a}$ is the probability with which agent i plays action $a \in A_i$ [26].

We also denote by Δ_i the space of strategies over A_i, i.e. vectors $\mathbf{x}_i$ such that $0 \leq x_{i,a} \leq 1$, for each $a \in A_i$, and $\sum_{a \in A_i} x_{i,a} = 1$ for each $i \in N$. By the previous definition every mixed strategy for agent i belongs to Δ_i.

Given a mixed strategy $\mathbf{x}_i$, for player i, the subset of actions that are assigned positive probability by $\mathbf{x}_i$ is called the support of $\mathbf{x}_i$.

Definition 2.1.5. The support of a mixed strategy $\mathbf{x}_i$ for a player i is the set of pure strategies $\{a_j | x_i(a_j) > 0\}$ [26].

Note that a pure strategy is a special case of a mixed strategy, in which the support is a single action. At the other end we have totally mixed strategies: a *totally mixed strategy* for player i is a mixed strategy in which the player assigns a strictly positive probability to every action $a \in A_i$.

2.1.4 Expected Utility

The payoff matrix defines the utility only for the special case of pure-strategy profiles, i.e. given a pure strategy profile $\mathbf{x} = (a_1, ..., a_k)$, where a_i is the action played by agent i, the utility outcome for each player is given by $U_i(a_1, ..., a_k) = U_i(\mathbf{x})$. However the generalization to mixed strategies is

straightforward, and relies on the basic notion of *expected utility*. To compute the expected utility for player i we first compute the probability of reaching each outcome given the strategy profile, and then we compute the average of the payoffs of the outcomes, weighted by the probabilities of each outcome. Formally, we define the expected utility as follows

Definition 2.1.6. Given a strategic–form game (N, A, U), the expected utility EU_i for agent i of the mixed strategy profile $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is defined as [26]:

$$EU_i(\mathbf{x}) = \sum_{(a_1,\ldots,a_k)\in A_1\times\cdots\times A_k} U_i(a_1, \ldots, a_k) \prod_{i\in N} x_{i,a_i}$$

2.1.5 Correlated strategies

Up to now it has been assumed that the players choose their strategies independently of each other, i.e. there is no communication. However, if coordination and communication are allowed among agents it is possible to choose a strategy profile which is "favourable" to each agent.

Definition 2.1.7. A correlated strategy is a probability distribution over the set of pure strategy profiles $a \in A$.

		agent 2	
		a_4	a_5
agent 1	a_1	a_{11}, b_{11}	a_{12}, b_{12}
	a_2	a_{21}, b_{21}	a_{22}, b_{22}
	a_3	a_{31}, b_{31}	a_{32}, b_{32}

Table 2.1: A generic bimatrix game where there are three actions available for the first agent and two for the second.

Remark that we can consider a mixed strategy for agent i a probability distribution over the set A_i, hence it is obvious that each mixed strategy profile is also a correlated strategy. On the other hand there are correlated strategies that can not be expressed as mixed strategies. For example consider the generic two–players game described in Table 2.1. Here the mixed strategy $\mathbf{x}_1 = (\frac{1}{2}\mathsf{a}_1, \frac{1}{2}\mathsf{a}_2, \frac{1}{2}\mathsf{a}_3)$, $\mathbf{x}_2 = (\frac{1}{2}\mathsf{a}_4, \frac{1}{2}\mathsf{a}_5)$ is equivalent to the correlated strategy that assigns to every pure strategy profile the same probability: $P((a_i, a_j)) = \frac{1}{6}$ for every $i \in \{1, 2, 3\}$ and $j \in \{4, 5\}$. However if we consider the correlated strategy $P((a_1, a_4)) = P((a_2, a_5)) = \frac{1}{2}$, $P((a_i, a_j)) = 0$ for

every other pure strategy, there is no mixed strategy profile equivalent to this correlated strategy.

If correlated strategies are allowed then the characterization of the Pareto frontier becomes a trivial task.

Proposition 2.1.8. *Given a strategic–form game* (N, A, U) *if all the agents in* N *can play correlated strategies then the Pareto frontier, in the utility* n*–dimensional space, is the convex combination of consecutive points in the set* $\{(u_1, \ldots, u_n) : u_i = U_i(a_1, \ldots, a_n), \forall (a_1, \ldots, a_n) \in A \text{ Pareto dominant}\}$

2.1.6 Bimatrix games

A particular case of strategic–form game is the *bimatrix game*, that is a game with only two players. In this case the game is defined by two payoff matrices, one for each player. In order to use a lighter notation, we can make the following assumptions:

- we will refer the utility matrices U_1, U_2 as A, B respectively, in order to remove players indices;
- the set of actions for the first agent is $A_1 = \{a_1, ..., a_m\}$ and the set of actions for the second agent is $A_2 = \{a_1, ..., a_n\}$, hence $A, B \in \mathbb{R}^{m \times n}$;
- actions will be addressed by their indices rather than their name, therefore, given a mixed strategy for the first player $\mathbf{x}_1$, the probability with which the agent plays action a_1 is denoted by x_{11} in place of x_{1,a_1}.

In a bimatrix game, given a strategy profile $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$, we can define the expected utility in a more fancy way using the definition of vector–matrix product. Hence, the expected utility is denoted by

$$EU_1(\mathbf{x}) = \mathbf{x}_1^T A \mathbf{x}_2 \tag{2.1}$$

$$EU_2(\mathbf{x}) = \mathbf{x}_1^T B \mathbf{x}_2 \tag{2.2}$$

A generic bimatrix game can be represented through a matrix whose element are pairs $(a_{ij}, b_{ij}) \in \mathcal{R}^2$, where a_{ij} and b_{ij} are the utility outcomes for the first and the second agent respectively, when the first agent plays action a_i and the second agent plays action a_j. An example of a bimatrix game is shown in Table 2.1.

There are particular cases in which the number of best responses to a strategy exceeds the size of the support of the strategy, we call this kind of games *degenerate* games. As these games represent particular cases, in general we are interested in *non–degenerate* games.

Definition 2.1.9. A bimatrix game is non–degenerate if and only if for every strategy $\mathbf{x}_1$ of the first agent, $|\text{supp}(\mathbf{x}_1)|$ is at least the number of pure best responses to $\mathbf{x}_1$, and for every strategy $\mathbf{x}_2$ of the second agent, $|\text{supp}(\mathbf{x}_2)|$ is bigger than or equal to the number of pure best responses to $\mathbf{x}_2$ [26].

In a degenerate game, Definition 2.1.9 is violated, for example if there is a pure strategy that has two pure best responses.

2.2 Solution Concepts

The problem of choosing the best strategy profile is not simple, as what is best for one agent might not be good for another. As such, different solution concepts have been proposed. In this section we will introduce three solution concepts: the *Pareto efficiency*, the *Nash equilibrium* and the *strong Nash equilibrium*.

2.2.1 Pareto efficiency

In general, given a strategic–form game, it is not possible to identify the single best outcome, as a state of the world that is good, i.e. give a high payoff, for one agent may be bad for the others. However there are situations in which one outcome is better than another for every agent. For example, in a bimatrix game the outcome $(10, 3)$ is better than $(9, 2)$ for both agents, and is better than $(9, 3)$, for the first agent. We formalize this intuition in the following definition.

Definition 2.2.1. In a strategic form game (N, A, U), given two strategy profiles $\mathbf{x}$ and $\mathbf{x}'$, the strategy $\mathbf{x}'$ is said to be (weakly) *Pareto dominated* by $\mathbf{x}$ if $EU_i(\mathbf{x}) \geq EU_i(\mathbf{x}')$ for each $i \in N$, and $EU_i(\mathbf{x}) > EU_i(\mathbf{x}')$ for at least one $i \in N$, in this case we say $\mathbf{x}' <_P \mathbf{x}$ [26].

The previous definition introduces a binary relation $<_P$ which is a partial order over the space of strategy profiles. Using this relation we can define the Pareto efficiency as follows:

Definition 2.2.2. A strategy $\mathbf{x}$ is *Pareto optimal*, or strictly *Pareto efficient* if there does not exists a strategy $\mathbf{x}' \neq \mathbf{x}$ such that $\mathbf{x} <_P \mathbf{x}'$ [26].

The relation $<_P$, that is the Pareto domination, allows us to state that some strategies are definitely better than others. In fact, given two strategy profiles $\mathbf{x}$ and $\mathbf{x}'$, if $\mathbf{x} <_P \mathbf{x}'$ we are sure that the strategy $\mathbf{x}'$ at least one agent

has an higher payoff and the other agents does not have a lower outcome than in $\mathbf{x}$. Nevertheless the Pareto dominance relation induces only a partial order over the strategy profiles, hence we cannot generally identify a single best outcome; instead, we may have a set of non–comparable optima.

We can relax the Pareto optimality requirement introducing the concept of weak Parteo efficiency. A strategy is weakly Pareto efficient if there are no other strategies whose outcome would cause every individual to gain.

Definition 2.2.3. A strategy profile $\mathbf{x}$ is weakly *Pareto efficient* if there does not exists a strategy profile $\mathbf{x}'$ such that $EU_i(\mathbf{x}') > EU_i(\mathbf{x}), \forall i$ [26].

Weak Pareto-optimality is weaker than strong Pareto optimality in the sense that any strategy which is a strong Pareto optimum is also a weak Pareto optimum, conversely a weak Pareto optimum allocation is not necessarily a strong Pareto optimum.

The problem of verifying whether a given strategy profile is weakly Pareto efficient is in a well known complexity class as stated in the following proposition.

Proposition 2.2.4. *Given a strategic–form game (N, A, U) and a strategy profile $\mathbf{x}$, the problem of verifying whether $\mathbf{x}$ is weakly Pareto efficient is in $\mathcal{P}$ [12].*

The *Pareto frontier* is the set of all strategies which are Pareto optimal. In the utility k–dimensional space, where k is the number of agents, the Pareto frontier is the set $P = \{u \in \mathbb{R}^k \mid u = (EU_1(\mathbf{x}), ..., EU_k(\mathbf{x})) \wedge \mathbf{x}$ is Pareto optimal $\}$.

In a generic strategic form game (N, A, U) we can state the problem of finding Pareto optimal strategy profiles as a nonlinear multi–objective optimization problem [19] as follows:

$$\max \ (EU_1(\mathbf{x}_1, \ldots, \mathbf{x}_n), \ldots, EU_n(\mathbf{x}_1, \ldots, \mathbf{x}_n))^T \tag{2.3}$$

$$\text{subject to: } \mathbf{x}_i^T \cdot \mathbf{1} - 1 = 0, \forall i \in N \tag{2.4}$$

$$\mathbf{x}_i \geq \mathbf{0}, \forall i \in N \tag{2.5}$$

Here through Constraint 2.4 and Contraint 2.5 we state that each vector $\mathbf{x}_1$ is a probability distribution over the set of actions A_i, hence the vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is a valid strategy profile for n players. The set of points that solve the Problem 2.3–2.5 denotes the Pareto frontier.

2.2.2 Nash equilibrium

The central solution concept in game theory is the Nash equilibrium. Informally, a strategy profile is a NE if no player can improve his payoff by unilaterally changing his or her strategy.

In order to define the Nash equilibrium, first of all consider the case in which the strategy of the agents other than i is known in advance, let say $\mathbf{x}_{-i}$, a utility–maximizing agent i would face the problem of determining his best response.

Definition 2.2.5. Agent i's *best response* to the strategy profile $\mathbf{x}_{-i}$ is a mixed strategy $\mathbf{x}_i^* \in \Delta_i$ such that $EU_i(\mathbf{x}_i^*, \mathbf{x}_{-i}) \geq EU_i(\mathbf{x}_i, \mathbf{x}_{-i})$ for all strategies $\mathbf{x}_i \in \Delta_i$ [26].

The best response is not necessarily unique. Indeed, except in the extreme case in which there is a unique best response that is a pure strategy, the number of best responses is always infinite. When the support of a best response $\mathbf{x}^*$ includes two or more actions, the agent must be indifferent among them otherwise, the agent would prefer to reduce the probability of playing at least one of the actions to zero. But thus any mixture of these actions must also be a best response. Similarly, if there are two pure strategies that are individually best responses, any mixture of the two is necessarily also a best response. In general an agent will not know what strategies the other players will adopt. Thus, the notion of best response is not a solution concept, however we can leverage the idea of best response to define what is arguably the most central notion in noncooperative game theory, the Nash equilibrium.

Definition 2.2.6. A strategy profile $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is said to be a *Nash equilibrium* if, for all agents i, $\mathbf{x}_i$ is a best response to $\mathbf{x}_{-i}$ [26].

However when we speak about Nash equilibrium we always refer to the notion of weak Nash equilibrium.

Definition 2.2.7. A strategy profile $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ is a *(weak) Nash Equilibrium* i, $EU_i(\mathbf{x}_i, \mathbf{x}_{-i}) \geq EU_i(\mathbf{x}_i', \mathbf{x}_{-i})$ for every $\mathbf{x}_i' \in \Delta_i$, for all agents i and for all strategies $\mathbf{x}_i' \neq \mathbf{x}_i$ [26].

Note that if mixed strategies are allowed, then every game with a finite number of players in which each player can choose from finitely many pure strategies has at least one Nash equilibrium.

Finally notice that Nash equilibria are resilient to unilateral deviations, as by Definition 2.2.7 no agent can improve his utility by changing his strategy. Therefore the Nash equilibrium is a stable solution concept under the hypothesis of noncooperative game.

2.2.3 Finding Nash equilibria

The equilibrium computation in non–cooperative games is one of the main topics in artificial intelligence and computer science. However the issue of characterizing the complexity of computing a sample Nash equilibrium is tricky. No known reduction exists from this problem to a decision problem which is $\mathcal{NP}$–complete, nor has the problem been shown to be easier [26]. Current knowledge about the complexity of computing a sample Nash equilibrium thus relies on another complexity class that describes the problem of finding a solution which always exists. This class is called $\mathcal{PPAD}$, which stands for "polynomial parity argument, directed version". The problem of searching for a Nash equilibrium is $\mathcal{PPAD}$–complete [8] even with two agents [5]. Many algorithms has been designed to compute Nash equilibria: two–agents games can be solved by linear complementary programming [17], support enumeration [22], or mixed–integer programming [24]. With more than two agents, classical methods based on a nonlinear complementarity programing, simplicial subdivision, or homotopy can be used [26], as can support enumeration [28].

Our interest is toward the general problem of finding a Nash equilibrium in a strategic–form game with n players and its representation through a nonlinear feasibility problem [26].

Formulation 2.2.8. The problem of finding a Nash equilibrium can be formulated through a nonliear feasibility problem as:

$$v_i - \sum_{a_{-i} \in A_{-i}} U_i(a_i, a_{-i}) \cdot \prod_{\substack{j \in N: \\ j \neq i}} x_j(a_j) \geq 0 \qquad \forall i \in N, a_i \in A_i \tag{2.6}$$

$$x_i(a_i) \cdot \left(v_i - \sum_{a_{-i} \in A_{-i}} U_i(a_i, a_{-i}) \cdot \right.$$

$$\left. \cdot \prod_{\substack{j \in N: \\ j \neq i}} x_j(a_j) \right) = 0 \qquad \forall i \in N, a_i \in A_i \tag{2.7}$$

$$x_i(a_i) \geq 0 \qquad \forall i \in N, a_i \in A_i \tag{2.8}$$

$$\sum_{a_i \in A_i} x_i(a_i) = 1 \qquad \forall i \in N \tag{2.9}$$

Here v_i is the expected utility of agent i. Constraints (2.6) force the expected utility v_i of agent i to be non–smaller than the expected utility given by every action a_i available to agent i. Constraints (2.7) force the expected utility v_i of agent i to be equal to the expected utility given by every action a_i that is

played with strictly positive probability by agent i. Constraints (2.8) force each probability $x_i(a_i)$ to be non–smaller than zero. Constraints (2.9) force each vector of probability $\mathbf{x}_i$ to sum to one.

2.2.4 Strong Nash equilibrium

The *strong Nash equilibrium* concept strengthens the Nash equilibrium by requiring the strategy profile to be resilient also tu multilateral deviations, including deviations in the grand coalition which contains all the agents [1], i.e. it is a Nash equilibrium in which no coalition, taking the actions of its complements as given, can cooperatively deviate in a way that benefits all of its members. While the Nash concept of stability defines equilibrium only in terms of unilateral deviations, strong Nash equilibrium allows for deviations by every conceivable coalition, including the grand coalition. In this case two concepts are combined: in order to be a strong Nash equilibrium a strategy profile has to be a Nash equilibrium and it has to be weakly Pareto efficient over the space of all the strategy profiles, including the mixed ones, for each possible coalition, given the strategies of the agents outside the coalition.

In order to define the strong Nash equilibrium, we introduce some notations.

Notation 2.2.9. Consider a strategic–form game (N, A, U). Let $\mathcal{C} = \{C : C \in \mathcal{P}(N), \mid C \mid \geq 2\}$ be the set of non–singleton, non–empty coalitions and let $C \in \mathcal{C}$ be a coalition. We denote a_C a profile of actions, i.e. a pure strategy profile, for each agent i that is a member of C and by $a_C(i)$ the action of agent i in a_C. In a similar way we denote by a_{-C} a profile of actions for the agents that are not members of C and by $a_{-C}(j)$ the action of agent j in a_{-C}. Finally we denote by $\mathbf{x}_C$ a strategy profile for the members of C and by $\mathbf{x}_{-C}$ a strategy profile for the agents that are not members of C.

Note that by the previous statements, given a coalition C, every profile of actions $(a_1, \ldots, a_n) \in A$ can be partitioned as $(a_C, a_{-c}) \in A$ and, in the same way, every mixed strategy profile $\mathbf{x}$ can be partitioned as $\mathbf{x} = (\mathbf{x}_C, \mathbf{x}_{-C})$.

Notation 2.2.10. Let $C \in \mathcal{C}$ be a coalition of game (N, A, U), where $N = \{1, \ldots, n\}$, $A = A_1 \times \cdots \times A_n$, $U = \{U_1, \ldots, U_n\}$, we denote the game (C, A_C, U_C) as a game with players $j \in C$, where $A_C = \prod_{j \in C} A_j$ and U_C is the set of the utility arrays for the agents in C such that $U_{C,j} = \sum_{a_{-C} \in A_{-C}} U_j(a_C, a_{-C}) \prod_{j \notin C} x_j(a_j)$ is the utility of agent j, when the agents in the coalition C play actions a_C

Definition 2.2.11. A strategy profile $\mathbf{x}$ is a strong Nash equilibrium of game (N, A, U), if [1] for every possible coalition $C \in \mathcal{C}$ there does not exists any strategy $\mathbf{x}_C$ such that $EU_i(\mathbf{x}_C, \mathbf{x}_{-C}) > EU_i(\mathbf{x}), \forall i \in N$

The Definition 2.2.11 immediately implies that any strong equilibrium is both weakly Pareto efficient and a Nash equilibrium. Indeed, if a coalition C deviates from its strategy $\mathbf{x}_C$ in some strong Nash equilibrium $\mathbf{x}$, then it cannot improve the earning of all its players at the same time if the rest of the players maintain its strategy $\mathbf{x}_{-C}$ of $\mathbf{x}$. This equilibrium is stable with regard to the deviation of any coalition.

We can now formulate the following Lemma, whose proof is straightforward given the previous definition and considerations.

Lemma 2.2.12. *A strategy profile* $\mathbf{x}$ *of game* (N, A, U) *is a Strong Nash equilibrium if all the following conditions hold:*

1. $\mathbf{x}$ *is a Nash equilibrium of game* (N, A, U)*,*
2. *for every possible coalition* $C \in \mathcal{C}$*, the strategy profile* $\mathbf{x}_C$*, such that* $\mathbf{x} = (\mathbf{x}_C, \mathbf{x}_{-C})$*, is a weakly Pareto efficient strategy of game* (C, A_C, U_C)*.*

Finally notice that even though every strategic–form game has at least a Nash equilibrium [26], the same existence result does not hold for strong Nash equilibria, in fact there are games in which such equilibrium does not exist, even in mixed strategies.

Lemma 2.2.13. *A strong Nash equilibrium may not exist in a strategic–form game.*

		agent 2	
		a_3	a_4
agent 1	a_1	3,3	0,5
	a_2	5,0	1,1

Table 2.2: Utility matrix of Prisoner's dilemma

Proof. Consider the strategic–form game in Table 2.2 (prisoners dilemma), this game has a unique Nash equilibrium that is the strategy (a_2, a_4). However this strategy profile is weakly Pareto dominated by (a_1, a_3), therefore there are no strategy profiles resilient to both unilateral and multilateral deviations, thus in the prisoners dilemma game a strong Nash equilibrium does not exist, even allowing mixed strategies. □

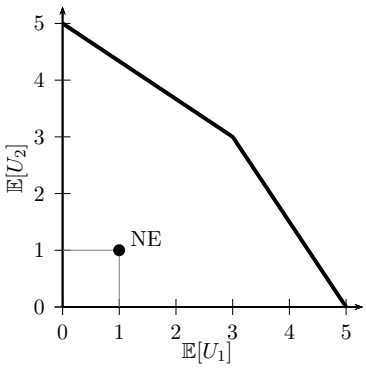

Figure 2.1: The Pareto frontier of the game described in Table 2.2.

2.2.5 Finding strong Nash equilibria

The problem of finding a strong Nash equilibrium in a strategic–form game has shown to be $\mathcal{NP}$–complete when the number of agents is a constant [12]. In general a strong Nash equilibrium must be simultaneously a Nash equilibrium and a solution of multiple non–convex optimization problems [15]. This makes even the derivation of necessary and sufficient equilibrium constraints in mathematical programming fashion a difficult and currently open task.

Some results have been proven about the computation of pure–strategy SNEs in specific classes of games, e.g., congestion games [16, 14, 23, 15], connection games [9], maxcut games [13], and continuous games [20]. The only prior algorithm that works also for mixed strategies is very recent [12]. It is only for two–agent games. It is a special kind of tree search algorithm, and at each node it calls an $\mathcal{NP}$–complete oracle, a variation of *MIP Nash* [24], that returns an Nash equilibrium (if one exists) in a given subspace of the agents' utilities and then verifies whether the returned Nash equilibrium is a strong Nash equilibrium. It has been proven that the problem of verify whether a strategy profile is a strong Nash equilibrium is in $\mathcal{P}$ [12].

2.3 Nonlinear Programming

In mathematics, nonlinear programming is the process of solving an optimization problem defined by a system of equalities and inequalities, collectively termed constraints, over a set of unknown real variables, along with an objective function to be maximized or minimized, where some of the constraints or the objective function are nonlinear. As discussed in

the previous sections it is possible to find some particular strategies solving nonlinear programs, in particular the solutions of the Problem 2.3–2.5 are the Pareto efficient strategies and the solutions of the Problem 2.6–2.9 are the strategy profiles that are Nash equilibria. Notice that the first is a multi–objective optimization problem with linear constraints and nonlinear objective functions, while the latter is a feasibility problem with two nonlinear constraints: 2.6, 2.7. The Karush–Kuhn–Tucker method allows us to derive from a multi–objective optimization problem P a feasibility problem $K(x)$, such that if x is a solution for P then $K(x)$ is feasible. In other words we can write a set of necessary conditions for Pareto optimality.

2.3.1 Karush–Kuhn–Tucker Conditions

The Karush-Kuhn-Tucker conditions [19] are first order necessary conditions for a solution in nonlinear programming to be optimal. Allowing inequality constraints, the Karush–Kuhn–Tucker approach to nonlinear programming generalizes the method of Lagrange multipliers [2], which allows only equality constraints.

Remark that an optimization problem can be defined as the minimization (or maximization) of a function over a given set. When we have more than one objective function to be optimized simultaneously we are dealing with multi-objective optimization. A typical multi–objective optimization problem is the following [19]:

$$\begin{aligned} \text{Minimize } & (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_n(\mathbf{x}))^T \\ \text{subject to: } & g_i(\mathbf{x}) \leq 0 && \forall i = 1, ..., m \\ & h_i(\mathbf{x}) = 0 && \forall i = 1, ..., \ell \end{aligned}$$

where $f_i(\mathbf{x})$ are the objective functions, $g_j(\mathbf{z})$ are the inequality constraints and h_k are the equality constraints. Notice that, given a point $\bar{\mathbf{x}} \in \mathcal{F}$, if $g_i(\bar{x}) = 0$ the i-th inequality constraint is said to be active.

Consider a problem P as sated above, let $\bar{x} \in \mathcal{F}$, be a feasible solution, and let $I = \{i : g_i(\bar{\mathbf{x}}) = 0\}$, i.e. the set of indices of active constraints. Suppose that f and g_i for $i \in I$ are differentiable at $\bar{\mathbf{x}}$, that each g_i for $i \notin I$ is continuous at $\bar{\mathbf{x}}$, and that each h_i for $i = 1, ..., \ell$ is continuously differentiable at $\bar{\mathbf{x}}$. Further, suppose that $\nabla g_i(\bar{\mathbf{x}})$ for $i \in I$ and $\nabla h_i(\bar{\mathbf{x}})$ for $i = 1, ..., \ell$ are linearly independent. If $\bar{\mathbf{x}}$ solves problem P locally, there exists unique scalars λ_i for $i = 1, ..., k$, μ_i for $i \in I$ and ν_i for $i = 1, ..., \ell$ such

that[2]:

$$\sum_{i=1}^{k} \lambda_i \nabla f_i(\bar{\mathbf{x}}) + \sum_{i \in I} \mu_i \nabla g_i(\bar{\mathbf{x}}) + \sum_{i=1}^{l} \nu_i \nabla h_i(\bar{\mathbf{x}}) = 0 \tag{2.10}$$

$$\lambda_i > 0 \qquad \forall i = 1, ..., k \tag{2.11}$$

$$\mu_i \geq 0 \qquad \forall i \in I \tag{2.12}$$

In addition to the above assumptions, if each g_i for $i \notin I$ is also differentiable at $\bar{\mathbf{x}}$, the Karush–Kuhn–Tucker conditions can be written in the following equivalent form [2]:

$$\sum_{i=1}^{k} \lambda_i \nabla f_i(\bar{\mathbf{x}}) + \sum_{i \in I} \mu_i \nabla g_i(\bar{\mathbf{x}}) + \sum_{i=1}^{l} \nu_i \nabla h_i(\bar{\mathbf{x}}) = 0 \tag{2.13}$$

$$\mu_i g_i(\bar{\mathbf{x}}) = 0 \qquad \forall i = 1, \ldots, m \tag{2.14}$$

$$\lambda_i > 0 \qquad \forall i = 1, \ldots, k \tag{2.15}$$

$$\mu_i \geq 0 \qquad \forall i = 1, \ldots, m \tag{2.16}$$

Thus the Equation 2.13 is a group of $|\mathbf{x}|$ equations. The λ_i, μ_j, ν_k are called Karush-Kuhn-Tucker multipliers: λ_i is the weight of objective function f_i, μ_j is the weight of constraint g_j, and ν_k is the weight of constraint h_k.

Remark that the Karush-Kuhn-Tucker conditions in general are necessary but not sufficient conditions for the given optimization problem.

Chapter 3

Properties of strong Nash equilibria

In this chapter we want to define under which conditions a strong Nash equilibrium exists. Moreover, using this result, we want to characterize the smoothed–complexity of the problem of finding a strong Nash equilibrium.

Our analysis starts considering the class of bimatrix games, i.e. games with two players. In this class of games, as there is only one possible non–singleton coalition, a strong Nash equilibrium can be defined as a Nash equilibrium which is also Pareto dominant. Our approach is to join these two problems by means of necessary conditions and consider the properties of the resulting problem.

In the first section of this chapter we provide necessary conditions for a strategy to be Pareto dominant through the Karush–Kuhn–Tucker conditions, with a particular focus on the case of maximal–support strategies. In the second section we derive a set of necessary conditions for a mixed–strategy to be a Nash equilibrium. In the third section we join together the two sets of necessary conditions in order to derive the necessary conditions for a strategy to be a strong Nash equilibrium. Again, we focus on the maximal–support strategies that allow us to write the necessary conditions as a set of equality constraints. In the fourth section we show that if a strong Nash equilibrium exists, then the payoffs restricted to the actions in the support of the equilibrium must satisfy a restrictive property: they must lie on the same line in the agents' utility space. In the last section we show that the problem of finding a strong Nash equilibrium is in smoothed–$\mathcal{P}$, admitting a deterministic support–enumeration algorithm with smoothed polynomial running time.

3.1 Pareto optimality in bimatrix games

As stated in Section 2.2.1, given a strategic form game, the problem of finding Pareto optimal strategies is a nonlinear multi-objective optimization problem P, with objective functions (2.3) and constraints (2.4), (2.5). Moreover, as described in Section 2.3.1, given a multi-objective optimization problem P and a candidate solution x, we can derive a set of necessary conditions $K(x)$ such that if x is a solution for P then $K(x)$ is satisfiable. The set $K(x)$ is the set of Karush–Kuhn–Tucker conditions. In this section we focus on deriving the Karush–Kuhn–Tucker conditions for the Pareto optimality problem in a bimatrix game.

3.1.1 Necessary conditions

When we are dealing with a bimatrix game, i.e. a strategic form game with two players, the nonlinear multi-objective optimization problem P is the following:

$$max(EU_1(\mathbf{x}_1, \mathbf{x}_2), EU_2(\mathbf{x}_1, \mathbf{x}_2))^T \tag{3.1}$$

$$\text{subject to: } \mathbf{x}_1^T \cdot \mathbf{1} - 1 = 0 \tag{3.2}$$

$$\mathbf{x}_2^T \cdot \mathbf{1} - 1 = 0 \tag{3.3}$$

$$\mathbf{x}_1 \geq \mathbf{0} \tag{3.4}$$

$$\mathbf{x}_2 \geq \mathbf{0} \tag{3.5}$$

Remark that in a bimatrix game we call A and B the agents' payoff matrices. Moreover we denote by a_{ij} and b_{ij} the elements of A and B respectively. The dimensions of the sets of actions, $|A_1| = m$ and $|A_2| = n$, define the dimensions of the matrices: $A, B \in \mathcal{R}^m \times \mathcal{R}^n$. If we consider the Equation (2.1), the Equation (2.2) and the definition of the mixed strategies $\mathbf{x}_1 = (x_{11}, ..., x_{1m})$ and $\mathbf{x}_2 = (x_{21}, ..., x_{2n})$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are column vectors, we can write the utility functions as follows:

$$EU_1(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} \cdot x_{1i} \cdot x_{2j}$$

$$EU_1(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij} \cdot x_{1i} \cdot x_{2j}$$

It is easy to verify that the gradients of the utility functions are:

$$\nabla EU_1 = \begin{pmatrix} A\mathbf{x}_2 \\ A^T\mathbf{x}_1 \end{pmatrix}$$

$$\nabla EU_2 = \begin{pmatrix} B\mathbf{x}_2 \\ B^T\mathbf{x}_1 \end{pmatrix}$$

Considering the first agent's strategy $\mathbf{x}_1$, the gradient of the i–th inequality constraint $x_{1i} \geq 0$, where $i = 1, \ldots, m$, is a column vector with $m + n$ elements, where the i–th element is equal to one and all the others are zeros. Similarly, for the second agent, the gradient of the j–th inequality constraint $x_{2j} \geq 0$, for $j = 1, \ldots, n$, is a column vector with $m+n$ elements, where the $m + j$–th element is equal to one and all the others are zeros. Therefore the sum $\sum_{i \in I} \mu_i \nabla g_i(\bar{\mathbf{x}})$ has as result a column vector μ such that μ_i is the i th element of μ.

The gradient of the equality constraint $h_1(\mathbf{x}) = \mathbf{x}_1^T \cdot \mathbf{1} - 1 = 0$ is a column vector of $m + n$ elements, with the first m elements equal to one and the others equal to zero. Conversely the gradient of $h_2(\mathbf{x}) = \mathbf{x}_1^T \cdot \mathbf{1} - 1 = 0$ is a column vector with m zeros and n ones.

Notice that $\nabla g_i(\mathbf{x})$, for all $i \in I$, $\nabla h_1(\mathbf{x})$ and $\nabla h_2(\mathbf{x})$, are always linearly independent as required in Section 2.3.1. Remark that I is the set of indices of active inequality constraints, i.e. $I = \{i : g_i(\mathbf{x}) = 0\}$, hence the set of active inequality constraints depends on the supports of agents' strategies. At most there can be $m - 1$ active inequality constraints for the first agent, and $n - 1$ for the second.

Replacing the previous definitions in the generic Problem (2.13)–(2.16), we write the Karush–Kuhn–Tucker conditions associated to the problem of Pareto optimality in bimatrix games:

$$\lambda_1 \begin{pmatrix} A\mathbf{x}_2 \\ A^T\mathbf{x}_1 \end{pmatrix} + \lambda_2 \begin{pmatrix} B\mathbf{x}_2 \\ B^T\mathbf{x}_1 \end{pmatrix} + \mu + \nu_1 \begin{pmatrix} \mathbf{1}_m \\ \mathbf{0}_n \end{pmatrix} + \\ + \nu_2 \begin{pmatrix} \mathbf{0}_m \\ \mathbf{1}_n \end{pmatrix} = 0 \tag{3.6}$$

$$\mu_i \cdot x_{1i} = 0, \text{ for } i = 1, \ldots, m \tag{3.7}$$

$$\mu_{m+j} \cdot x_{2j} = 0, \text{ for } j = 1, \ldots, n \tag{3.8}$$

$$\mu_i \geq 0, \text{ for } i = 1, \ldots, m + n \tag{3.9}$$

$$\lambda_1 > 0 \tag{3.10}$$

$$\lambda_2 > 0 \tag{3.11}$$

Notice that the Pareto optimality problem is a maximization problem, however it is always possible to obtain an equivalent minimization problem like the one described is Section 2.3.1. By trivial mathematics it is possible to

show that the above conditions, derived from the maximization problem, are the same of those derived from the equivalent minimization problem.

We can write the Equation (2.13) by means of scalar equations as follows:

$$\begin{aligned}
\lambda_1(a_{11}x_{21} + \ldots + a_{1n}x_{2n}) + \lambda_2(b_{11}x_{21} + \ldots + b_{1n}x_{2n}) + \\
+\mu_1 + \nu_1 = 0 \\
\vdots \\
\lambda_1(a_{m1}x_{21} + \ldots + a_{mn}x_{2n}) + \lambda_2(b_{m1}x_{21} + \ldots + b_{mn}x_{2n}) + \\
+\mu_m + \nu_1 = 0 \\
\lambda_1(a_{11}x_{11} + \ldots + a_{m1}x_{1m}) + \lambda_2(b_{11}x_{11} + \ldots + b_{m1}x_{1m}) + \\
+\mu_{m+1} + \nu_2 = 0 \\
\vdots \\
\lambda_1(a_{1n}x_{11} + \ldots + a_{mn}x_{1m}) + \lambda_2(b_{1n}x_{1n} + \ldots + b_{mn}x_{1m}) + \\
+\mu_{m+n} + \nu_2 = 0
\end{aligned} \tag{3.12}$$

Notice that the sum $a_{i1}x_{21} + \ldots + a_{in}x_{2n}$ is the result of the product between the i–th row of the matrix A and the vector $\mathbf{x}_2$. The same holds for the sum $a_{1j}x_{11} + \ldots + a_{mj}x_{1m}$, which is the product of the j–th column of A and $\mathbf{x}_1$. Therefore, if we denote by:

- $\mathbf{r}_{Ai}$ the i–th row of A,
- $\mathbf{r}_{Bi}$ the i–th row of B,
- $\mathbf{c}_{Aj}$ the j–th column of A,
- $\mathbf{c}_{Bj}$ the j–th column of B;

we can use a more compact notation:

$$a_{i1}x_{21} + \ldots + a_{in}x_{2n} = \mathbf{r}_{Ai}\mathbf{x}_2 \tag{3.13}$$
$$b_{i1}x_{21} + \ldots + b_{in}x_{2n} = \mathbf{r}_{Bi}\mathbf{x}_2 \tag{3.14}$$
$$a_{1j}x_{11} + \ldots + a_{mj}x_{1m} = \mathbf{x}_1^T\mathbf{c}_{Aj} \tag{3.15}$$
$$b_{1j}x_{11} + \ldots + b_{mj}x_{1m} = \mathbf{x}_1^T\mathbf{c}_{Bj} \tag{3.16}$$

With this notation we can write the Equations (3.12) as follows:

$$\lambda_1\mathbf{r}_{Ai}\mathbf{x}_2 + \lambda_2\mathbf{r}_{Bi}\mathbf{x}_2 + \mu_1 + \nu_1 = 0, \text{ for } i = 1, \ldots, m \tag{3.17}$$
$$\lambda_1\mathbf{x}_1^T\mathbf{c}_{Aj} + \lambda_2\mathbf{x}_1^T\mathbf{c}_{Bj} + \mu_{m+1} + \nu_2 = 0, \text{ for } j = 1, \ldots, n \tag{3.18}$$

Observe that $\mathbf{r}_{Ai}\mathbf{x}_2$ is the utility outcome for the first agent when he plays his i–th action, given the strategy of the second agent, and $\mathbf{x}_1^T\mathbf{c}_{Bj}$ is the

utility outcome for the second agent when he plays his j–th action, given the first agent's strategy.

However we have to keep in mind that the Karush–Kuhn–Tucker conditions are first order necessary, and in general not sufficient, conditions for the local Pareto optimality. Therefore, given a bimatrix game and a strategy profile $\mathbf{x}$, if conditions (3.17),(3.18), (3.7)–(3.11) are satisfiable there is no guarantee that $\mathbf{x}$ is Pareto optimal. Conversely this problem is not satisfiable, then we are sure that $\mathbf{x}$ is Pareto dominated.

3.1.2 Maximal support strategies

It is interesting to notice that, if we are dealing with a maximal support strategy profile $\mathbf{x}^*$, in the problem P of determine whether $\mathbf{x}^*$ is Pareto efficient, as stated in Section 2.3.1, there are no active inequality constraints. In fact by definition each action of a strategy with maximal support is played with positive, non–zero, probability. Therefore in the Equations (3.17), (3.18) we have $\mu_i = 0$ for all $i = 1, ..., m + n$. As usual remark that, according to the Karush–Kuhn–Tucker conditions, if $\mathbf{x}^*$ is Pareto efficient, then the Problem (3.6)–(3.11) is feasible. Considering only maximal support strategies this problem looses some degrees of freedom, and since ν_1, ν_2 are arbitrary constants we can write the Karush–Kuhn–Tucker conditions as follows:

$$\lambda_1 \mathbf{r}_{Ai} \mathbf{x}_2^* + \lambda_2 \mathbf{r}_{Bi} \mathbf{x}_2^* = \nu_1, \forall i = 1, ..., m \tag{3.19}$$

$$\lambda_1 \mathbf{x}_1^{*T} \mathbf{c}_{Aj} + \lambda_2 \mathbf{x}_1^{*T} \mathbf{c}_{Bj} = \nu_2, \forall j = 1, ..., n \tag{3.20}$$

$$\lambda_1, \lambda_2 > 0 \tag{3.21}$$

3.2 Mixed strategy Nash equilibrium conditions

In a bimatrix game, it is easy to show that a mixed strategy profile $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ is a mixed Nash equilibrium if and only if each action in the support of $\mathbf{x}_1$ is a best response to the strategy $\mathbf{x}_2$ and vice versa. Therefore if the second agent plays the strategy $\mathbf{x}_2$ the first agent has the same utility outcome for each action in $\text{supp}(\mathbf{x}_1)$ and there are no actions outside this set which provide higher utility.

Thanks to this property we can derive a set of necessary conditions for

a mixed strategy profile $\mathbf{x}$ to be a Nash equilibrium.

$$\begin{aligned} EU_1(a_i, \mathbf{x}_2) &= EU_1(a_j, \mathbf{x}_2) && \forall a_i, a_j \in \text{supp}(\mathbf{x}_1) \\ EU_2(\mathbf{x}_1, a_i) &= EU_2(\mathbf{x}_1, a_j) && \forall a_i, a_j \in \text{supp}(\mathbf{x}_2) \end{aligned}$$

Where $EU_1(a_i, \mathbf{x}_2)$ is the expected utility of the first agent if the action a_i is played, fixed the second agent's strategy $\mathbf{x}_2$.

Moreover notice that $EU_1(\mathbf{x}) = EU_1(a_i, \mathbf{x}_2)$ for each a_i in $\text{supp}(\mathbf{x}_1)$, and $EU_2(\mathbf{x}) = EU_2(\mathbf{x}_1, a_j)$ for each a_j in $\text{supp}(\mathbf{x}_2)$.

In general, the expected utility for the first agent, given a strategy for the second agent $\mathbf{x}_2$, when action a_i is played is:

$$EU_1(a_i, \mathbf{x}_2) = a_{i1}x_{21} + ... + a_{in}x_{2n}$$

considering the Equation (3.13) we can write:

$$EU_1(a_i, \mathbf{x}_2) = \mathbf{r}_{Ai}\mathbf{x}_2$$

The same holds for the second agent, in fact, by the previous definitions and by Equation (3.16), we have:

$$EU_2(\mathbf{x}_1, a_i) = \mathbf{r}_{Bi}\mathbf{x}_2$$

Therefore if a mixed strategy $\mathbf{x}$ is a Nash equilibrium, then necessarily:

$$\mathbf{r}_{Ai}\mathbf{x}_2 = \mathbf{r}_{Aj}\mathbf{x}_2 \qquad \forall i, j \in \{k : a_k \in \text{supp}(\mathbf{x}_1)\} \tag{3.22}$$

$$\mathbf{x}_1^T\mathbf{c}_{Bi} = \mathbf{x}_1^T\mathbf{c}_{Bj} \qquad \forall i, j \in \{k : a_k \in \text{supp}(\mathbf{x}_2)\} \tag{3.23}$$

Moreover if we consider a maximal–support mixed strategy $\mathbf{x}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*)$, then, by the previous considerations, it is a Nash equilibrium if and only if for both agents each action provides the same expected utility, given the strategy profile of the other agent, that is:

$$\mathbf{r}_{Ai}\mathbf{x}_2^* = \mathbf{r}_{Aj}\mathbf{x}_2^* \qquad \forall i, j \in \{1, ..., m\} \tag{3.24}$$

$$\mathbf{x}_1^{*T}\mathbf{c}_{Bi} = \mathbf{x}_1^{*T}\mathbf{c}_{Bj} \qquad \forall i, j \in \{1, ..., n\} \tag{3.25}$$

In a bimatrix game with m available actions for the first agent and n actions for the second, the above conditions are *necessary and sufficient* for a maximal–support mixed strategy profile to be a Nash equilibrium.

3.3 Strong Nash Necessary Conditions

Up to now we derived necessary conditions for a strategy to be Pareto optimal and a Nash equilibrium. Remark that, by the lemma 2.2.12, in a two players game a strategy is a strong Nash equilibrium if and only if it is a Nash equilibrium and it is Pareto dominant. Hence we can now derive a set of necessary conditions for a strategy to be a strong Nash equilibrium.

Let a strategy profile $\mathbf{x}$ be a strong Nash equilibrium, then it has to be a Nash equilibrium too and it satisfies conditions (3.22)–(3.23), moreover $\mathbf{x}$ is also Pareto efficient, hence the problem (3.17), (3.18), (3.7)–(3.11) is feasible. We can combine the two sets of conditions as follows:

$$\begin{aligned}
\mathbf{r}_{Ai}\mathbf{x}_2 &= \mathbf{r}_{Aj}\mathbf{x}_2 && \forall i,j \in \mathrm{supp}(\mathbf{x}_1)\\
\mathbf{x}_1^T\mathbf{c}_{Bi} &= \mathbf{x}_1^T\mathbf{c}_{Bj} && \forall i,j \in \mathrm{supp}(\mathbf{x}_2)\\
\lambda_1\cdot(\mathbf{r}_{Ai}\mathbf{x}_2)+\lambda_2\cdot(\mathbf{r}_{Bi}\mathbf{x}_2)+\mu_i+\nu_1 &= 0 && \forall i=1,...,m\\
\lambda_1\mathbf{x}_1^T\mathbf{c}_{Aj}+\lambda_2\mathbf{x}_1^T\mathbf{c}_{Bj}+\mu_{m+j}+\nu_2 &= 0 && \forall j=1,...,n\\
\mu_i &= 0 && \forall i \in \mathrm{supp}(\mathbf{x}_1)\\
\mu_{m+j} &= 0 && \forall j \in \mathrm{supp}(\mathbf{x}_2)\\
\boldsymbol{\mu} &\geq \mathbf{0}\\
\boldsymbol{\lambda} &> \mathbf{0}
\end{aligned}$$

Consider now a strategy $\mathbf{x}^*$ which is a mixed strong Nash equilibrium with maximal support, then in the previous problem every possible action for agent i is in $\mathrm{supp}(\mathbf{x}_i)$. Therefore the problem reduces to:

$$\begin{aligned}
\mathbf{r}_{Ai}\mathbf{x}_2^* - \mathbf{r}_{Aj}\mathbf{x}_2^* &= 0 && \forall i,j=1,...,m\\
\mathbf{x}_1^{*T}\mathbf{c}_{Bi} - \mathbf{x}_1^{*T}\mathbf{c}_{Bj} &= 0 && \forall i,j=1,...,n\\
\lambda_1\mathbf{r}_{Ai}\mathbf{x}_2^*+\lambda_2\mathbf{r}_{Bi}\mathbf{x}_2^* &= \nu_1 && \forall i=1,...,m\\
\lambda_1\mathbf{x}_1^{*T}\mathbf{c}_{Aj}+\lambda_2\mathbf{x}_1^{*T}\mathbf{c}_{Bj} &= \nu_2 && \forall j=1,...,n\\
\boldsymbol{\lambda} &> \mathbf{0}
\end{aligned}$$

It is easy to verify that the previous problem has a solution if and only if $\mathbf{r}_{Bi}\mathbf{x}_2^* = \mathbf{r}_{Bj}\mathbf{x}_2^*$ for all $i,j=1,...,m$ and $\mathbf{x}_1^{*T}\mathbf{c}_{Ai} = \mathbf{x}_1^{*T}\mathbf{c}_{Aj}$ for all $i,j=1,...,n$. As our concern is the satisfiability of the problem and not the problem itself, we can write the *strong Nash equilibrium necessary conditions*

for a maximal support strategy in a equivalent form as follows:

$$\mathbf{r}_{Ai}\mathbf{x}_2^* - \mathbf{r}_{Aj}\mathbf{x}_2^* = 0 \qquad \forall i,j = 1,...,m \tag{3.26}$$

$$\mathbf{x}_1^{*T}\mathbf{c}_{Bi} - \mathbf{x}_1^{*T}\mathbf{c}_{Bj} = 0 \qquad \forall i,j = 1,...,n \tag{3.27}$$

$$\mathbf{r}_{Bi}\mathbf{x}_2^* - \mathbf{r}_{Bj}\mathbf{x}_2^* = 0 \qquad \forall i,j = 1,...,m \tag{3.28}$$

$$\mathbf{x}_1^{*T}\mathbf{c}_{Ai} - \mathbf{x}_1^{*T}\mathbf{c}_{Aj} = 0 \qquad \forall i,j = 1,...,n \tag{3.29}$$

This is the set of strong Nash necessary conditions for a maximal support strategy in a bimatrix game. Notice that, by the transitivity and reflexivity of the equality relation, in order to solve this problem it is sufficient to satisfy $2(m+n)-4$ equations: $m+n-2$ equations from the Nash equilibrium conditions and $m+n-2$ equations from the Karush–Kuhn–Tucker conditions.

3.4 Games with mixed strategy strong Nash equilibrium

In this section we use the previous results and definitions to show properties of games which allow the existence of a mixed–strategy strong Nash equilibrium. At first we will focus our analysis on bimatrix games with two actions per agent, then we generalize our results. In particular we are able to show that if a game has a mixed–strategy strong Nash equilibrium, then the payoffs, restricted to the actions of the support of the equilibrium, have to satisfy a restrictive property: they must lie on the same line in agents' utility space.

3.4.1 Bimatrix games with two actions per agent

Let denote by $\mathcal{P}_M$ and by $\mathcal{P}_C$ the Pareto frontier when the agents play mixed and correlated strategies respectively. Obviously, points in $\mathcal{P}_C$ non–strictly dominate points in $\mathcal{P}_M$, given that mixed strategies constitute a subset of correlated strategies as stated in Section 2.1.5. In addition, we denote by $\mathcal{P}_M(S)$ and $\mathcal{P}_C(S)$ the Pareto frontiers in mixed and correlated strategies respectively restricted to the set of actions in S.

Theorem 3.4.1. *Given a non–degenerate two–agent game with two actions per agent, if there is a mixed–strategy strong Nash equilibrium, then* $\mathcal{P}_M \equiv \mathcal{P}_C$.

agent 1 \ agent 2	a_3	a_4
a_1	p_1, q_1	p_2, q_2
a_2	p_3, q_3	p_4, q_4

Table 3.1: A generic bimatrix game with two actions per agent.

Proof. Consider a non–degenerate game as shown in Table 3.1, assume that there are no dominant pure strategies and assume that there exists a mixed strategy $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ which is a strong Nash equilibrium. By these assumptions the support of $\mathbf{x}$ is the union of the two agents' sets of actions, hence $\mathbf{x}$ is a maximal support strategy. Remark that $\mathbf{x}_1$ and $\mathbf{x}_2$ are two–dimensional column vectors, such that $\mathbf{x}_1 = (x_{11}, x_{12})$ and $\mathbf{x}_2 = (x_{21}, x_{22})$. We can write the Problem (3.26)–(3.29), i.e. the strong Nash necessary conditions for a maximal support strategy profile, for this bimatrix game as follows:

$$
\begin{aligned}
x_{21} \cdot p_1 + x_{22} \cdot p_2 - x_{21} \cdot p_3 + x_{22} \cdot p_4 = 0 \\
x_{11} \cdot q_1 + x_{12} \cdot q_3 - x_{11} \cdot q_2 + x_{12} \cdot q_4 = 0 \\
x_{21} \cdot q_1 + x_{22} \cdot q_2 - x_{21} \cdot q_3 + x_{22} \cdot q_4 = 0 \\
x_{11} \cdot p_1 + x_{12} \cdot p_3 - x_{11} \cdot p_2 + x_{12} \cdot p_4 = 0
\end{aligned}
$$

To solve the problem first of all we factor the x_ij variables and then we explicit a value for x_{11} and x_{21}. The result is the following:

$$
\begin{aligned}
x_{21} \cdot (p_1 - p_3) = x_{22} \cdot (p_4 - p_2) \\
x_{11} \cdot (q_1 - q_2) = x_{12} \cdot (q_4 - q_3) \\
x_{21} \cdot (q_1 - q_3) = x_{22} \cdot (q_4 - q_2) \\
x_{11} \cdot (p_1 - p_2) = x_{12} \cdot (p_4 - p_3)
\end{aligned}
$$

We can safely assume that $p_1 \neq p_3$, $p_4 \neq p_2$, $p_4 \neq p_3$ and $p_1 \neq p_2$, and the analogous inequalities for the second agent. This assumption excludes only degenerate games, in fact if $p_1 = p_3$, as by assumption $x_{22} > 0$, then by the above equations we have $p_4 = p_2$ and therefore actions a_1 and a_2 are the same for the first agent. However in such a game every pure strategy of the second agent has two best responses, violating Definition 2.1.9, thus the resulting game is degenerate. Moreover if $p_1 = p_2$ then, as $x_{12} > 0$, necessarily $p_4 = p_3$, thus if $p_1 \neq p_3$ one action of the first player strictly dominates the other and the dominated action can be ignored as it will never be played. In this case the game is equivalent to a bimatrix game

with one action for the first player and two actions for the second. The same reasoning holds for the second agent. By the previous assumptions, it is easy to verify that this problem allows solutions if and only if:

$$\frac{q_4 - q_2}{p_4 - p_2} = \frac{q_1 - q_3}{p_1 - p_3} \tag{3.30}$$

$$\frac{q_1 - q_2}{p_1 - p_2} = \frac{q_4 - q_3}{p_4 - p_3} \tag{3.31}$$

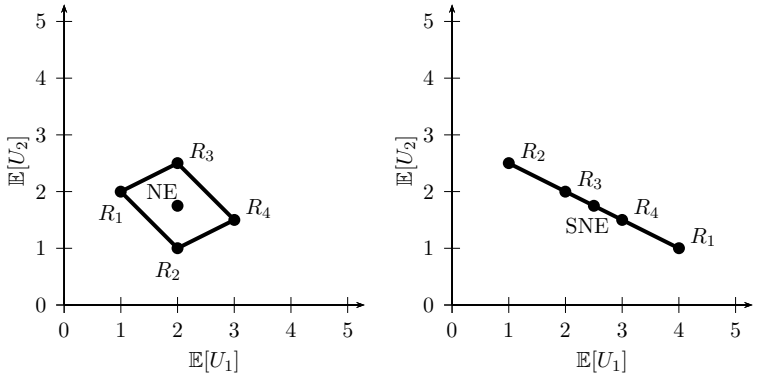

Figure 3.1: Examples used in the proof of Theorem 3.4.1.

We can give a simple geometric interpretation of Conditions 3.31–3.30. Call $R_i = (p_i, q_i)$. Each R_i is a point in the space $\mathbb{E}[U_1], \mathbb{E}[U_2]$. The above conditions state that:

- $\overline{R_1 R_2}$ is parallel to $\overline{R_3 R_4}$,
- $\overline{R_1 R_3}$ is parallel to $\overline{R_2 R_4}$,

and therefore R_1, R_2, R_3, R_4 are the vertices of a parallelogram, see Figure 3.1(a). Given that:

- a mixed strategy Nash equilibrium is strictly inside the parallelogram, being the convex combination of the vertices, see Figure 3.1(a)
- a strong Nash equilibrium has to be on a Pareto efficient edge since, if it is strictly inside the parallelogram, then it is Pareto dominated by some point on some edge, see Figure 3.1(a),

we have that R_1, R_2, R_3, R_4 must be aligned according to a line of the form $\phi \cdot \mathbb{E}[U_1] + (1-\phi) \cdot \mathbb{E}[U_2] = const$ with $\phi \in [0, 1]$, see, e.g., Figure 3.1(b). Thus, the combination of R_1, R_2, R_3, R_4 through every mixed–strategy profile lays

on the line connecting the two extreme vertices, e.g., in Figure 3.1(b), the extreme vertices are R_2 and R_1. It trivially follows that $\mathcal{P}_M \equiv \mathcal{P}_C$. □

The proof of the above theorem provides necessary conditions for a game to admit a mixed–strategy strong Nash equilibrium. These conditions are not sufficient. Indeed, we can show that only some alignments of R_1, R_2, R_3, R_4 lead to the existence of a strong Nash equilibrium.

Corollary 3.4.2. *The only alignments of R_1, R_2, R_3, R_4 leading to a mixed–strategy strong Nash equilibrium satisfy the following conditions:*

- *R_1 or R_4 is one extreme,*
- *moving from the previous extreme, the next vertex is R_4 or R_1,*
- *the next vertex is R_2 or R_3,*
- *R_3 or R_2 is the other extreme,*
- *the slope of the line where the R_i points lie is strictly negative.*

Proof. For alignments different from those considered in the corollary, it is not possible to have a mixed–strategy Nash equilibrium.

First of all consider the case in which the slope of the line where the R_i points lie is positive or equal to zero. It is immediate to notice that in this case there is only one pure strategy profile which Pareto dominates each other strategy, that is the upper–right edge of the segment connecting the R_i points. Also notice that this strategy is a Nash equilibrium, as no agent can improve his utility. Hence there exists a pure–strategy strong Nash equilibrium, but there can be no mixed–strategy Nash equilibria which are also Pareto efficient as, by assumption, there is a pure strategy which dominates every other. Therefore a mixed–strategy strong Nash equilibrium does not exist. An example of this situation is shown in Figure 3.2.

		agent 2	
agent 1		a_3	a_4
	a_1	$2,2\ (R_1)$	$3,3\ (R_2)$
	a_2	$1,1\ (R_3)$	$4,4\ (R_4)$

Table 3.2: Example of two–agent game which satisfies Conditions 3.31–3.30 but does not have a mixed–strategy strong Nash equilibrium.

Now, consider a generic bimatrix game with two actions per agent, in order to have a mixed–strategy strong Nash equilibrium, a mixed–strategy

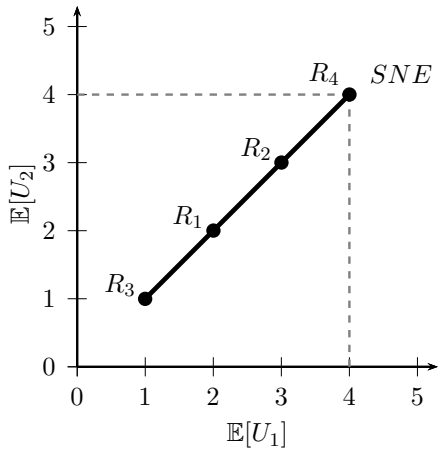

Figure 3.2: The Pareto frontier of the game described in Table 3.2.

Nash equilibrium has to exist. Hence, at first we need to determine under which conditions a mixed strategy $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$, whose support is such that $|\text{supp}(\mathbf{x}_1)| = |\text{supp}(\mathbf{x}_1)| = 2$, is a Nash equilibrium. If we consider the game described in Table 3.1, as $\mathbf{x}$ is a maximal–support strategy, by Conditions 3.24–3.25 we have that $\mathbf{x}$ is a Nash equilibrium if and only if:

$$x_{21} \cdot p_1 + x_{22} \cdot p_2 = x_{21} \cdot p_3 + x_{22} \cdot p_4$$
$$x_{11} \cdot q_1 + x_{12} \cdot q_3 = x_{11} \cdot q_2 + x_{12} \cdot q_4$$

Remark that by definition of mixed strategy we have $x_{12} = 1 - x_{11}$ and $x_{22} = 1 - x_{21}$. By trivial mathematics we obtain:

$$x_{21} = \frac{p_4 - p_2}{p_1 - p_2 - p_3 + p_4}$$
$$x_{11} = \frac{q_4 - q_3}{q_1 - q_2 - q_3 + q_4}$$

By definition of maximal–support strategy x_{11} and x_{21} have to be non–negative and smaller than one, hence the following inequalities have to hold:

$$0 < \frac{p_4 - p_2}{p_1 - p_2 - p_3 + p_4} < 1$$
$$0 < \frac{q_4 - q_3}{q_1 - q_2 - q_3 + q_4} < 1$$

Solving these inequalities we obtain two different sets of solutions which are conditions on the payoff matrices:

$$(p_1 > p_3 \wedge p_4 > p_2) \vee (p_1 < p_3 \wedge p_4 < p_2)$$
$$(q_1 > q_2 \wedge q_4 > q_3) \vee (q_1 < q_2 \wedge q_4 < q_3)$$

Combining these solutions we obtain four sets of conditions over the elements of the payoff matrices.

agent 1 \ agent 2	a_3	a_4
a_1	$3, 0$ (R_1)	$0, 3$ (R_2)
a_2	$1, 2$ (R_3)	$2, 1$ (R_4)

Table 3.3: Example of two–agent game with a mixed–strategy strong Nash equilibrium.

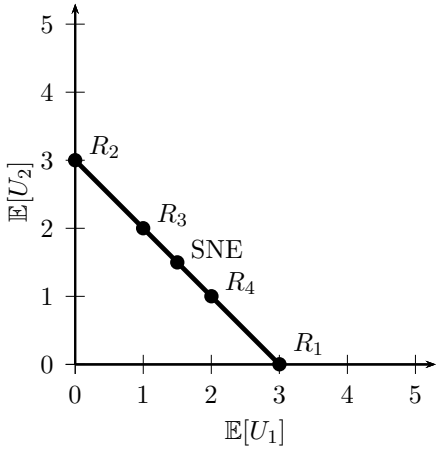

Figure 3.3: The Pareto frontier of the game described in Table 3.3.

Remark that the necessary condition for the existence of a mixed–strategy strong Nash equilibrium is that, in the utility space, each point $R_i = (p_i, q_i)$ has to lie on the same line, and we already showed that this line necessarily has a negative slope. As a consequence of the decreasing trend of this line, for each couple of point R_i, R_j their ordering over the $\mathbb{E}[U_1]$ axis is inverted over the $\mathbb{E}[U_2]$ axis, i.e. if $p_i < p_j$ then $q_i > q_j$ and vice versa. Consider now the set $p_1 > p_3 \wedge p_4 > p_2$, these conditions together with te alignment condition are not compatible with $q_1 > q_2 \wedge q_4 > q_3$, in fact

$$\begin{aligned} q_1 > q_2 &\Rightarrow p_1 < p_2 \\ q_4 > q_3 &\Rightarrow p_4 < p_3 \\ p_1 > p_3 \wedge p_1 < p_2 &\Rightarrow p_3 < p_2 \\ p_3 < p_2 \wedge p_4 > p_2 &\Rightarrow p_3 < p_4 \\ p_3 < p_4 \wedge p_3 > p_4 &\Rightarrow \bot \end{aligned}$$

In the same way it is possible to show that $p_1 < p_3 \wedge p_4 < p_2$ together with $q_1 < q_2 \wedge q_4 < q_3$ are not compatible with the alignment condition.

Therefore the only possible sets of conditions are:

$$p_1 > p_3 \wedge p_4 > p_2 \wedge q_1 < q_2 \wedge q_4 < q_3$$
$$p_1 < p_3 \wedge p_4 < p_2 \wedge q_1 > q_2 \wedge q_4 > q_3$$

As the R_i points lie on a negative slope line we can consider only the orderings over the $\mathbb{E}[U_1]$ axis:

$$p_1 > p_3 \wedge p_4 > p_2 \wedge p_1 > p_2 \wedge p_4 > p_3$$
$$p_1 < p_3 \wedge p_4 < p_2 \wedge p_1 < p_2 \wedge p_4 < p_3$$

It is easy to verify from this last set of conditions that the only alignments that guarantee the existence of a mixed–strategy strong Nash equilibrium are those described in Corollary 3.4.2. □

3.4.2 Generic bimatrix games

Now we extend the previous result to the case in which each agent has m actions.

Corollary 3.4.3. *Given a non–degenerate two–agent game with m actions per agent, if there is a mixed–strategy strong Nash equilibrium with $|supp(\mathbf{x}_1)| = |supp(\mathbf{x}_2)| = 2$, then $\mathcal{P}_M(supp(\mathbf{x}_1), supp(\mathbf{x}_2)) \equiv \mathcal{P}_C(supp(\mathbf{x}_1), supp(\mathbf{x}_2))$.*

Proof. Nash equilibrium constraints and Karush–Kuhn–Tucker conditions over the actions belonging to $\text{supp}(\mathbf{x}_1)$ and $\text{supp}(\mathbf{x}_2)$ are the same of the case with two actions per agent considered in the proof of Theorem 3.4.1, notice that additional constraints over the actions off the supports are not useful for the proof. Thus, restricting the game to the actions in $\text{supp}(\mathbf{x}_1)$ and $\text{supp}(\mathbf{x}_2)$, Theorem 3.4.1 holds and therefore $\mathcal{P}_M(\text{supp}(\mathbf{x}_1), \text{supp}(\mathbf{x}_2)) \equiv \mathcal{P}_C(\text{supp}(\mathbf{x}_1), \text{supp}(\mathbf{x}_2))$, see, e.g., Figure 3.4. □

We extend Corollary 3.4.3 to the case in which the supports have an arbitrary size.

Theorem 3.4.4. *Given a non–degenerate two–agent game with m actions per agent, if there is a mixed–strategy strong Nash equilibrium with $|supp(\mathbf{x}_1)| = |supp(\mathbf{x}_2)| = \overline{m}$, then $\mathcal{P}_M(supp(\mathbf{x}_1), supp(\mathbf{x}_2)) \equiv \mathcal{P}_C(supp(\mathbf{x}_1), supp(\mathbf{x}_2))$.*

Proof (sketch). The proof is similar to the proof of Theorem 3.4.1. Define $R_{i,j} = (U_1(i,j), U_2(i,j))$. By the Nash equilibrium constraints 3.22–3.23, and Karush–Kuhn–Tucker conditions 3.6–3.11, we have that:

		agent 2			
		a_5	a_6	a_7	a_8
agent 1	a_1	3,0	0,3	-5,-5	-5,-5
	a_2	1,2	2,1	-5,-5	-5,-5
	a_3	-5,-5	-5,-5	5,0	0,0
	a_4	-5,-5	-5,-5	0,0	0,5

Table 3.4: Example of two–agent game, with more than two actions per agent, with a mixed–strategy strong Nash equilibrium.

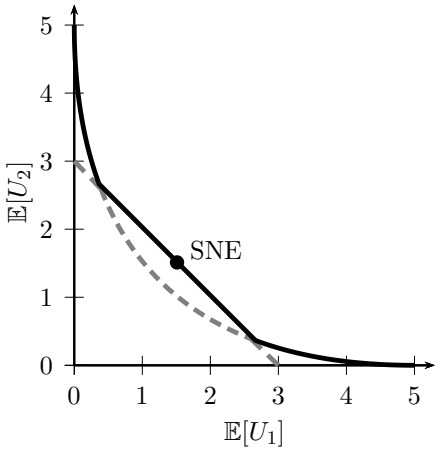

Figure 3.4: The Pareto frontier of the game described in Table 3.4.

- the convex combinations, with the same weights, of points $\{R_{i,j} : \forall i\}$ for all j must be the same (i.e., the convex combination of the elements of each column of the bimatrix must be the same),
- the convex combinations, with the same weights, of points $\{R_{i,j} : \forall j\}$ for all i must be the same (i.e., the convex combination of the elements of each row of the bimatrix must be the same),

in addition we have that:

- each convex combination is strictly inside the polygon whose vertices are points $R_{i,j}$, the combination not being degenerate,
- the combination must be on a Pareto efficient edge, it would be Pareto dominated otherwise,

and therefore all the vertices must be aligned. In this way, the combination of all the points $R_{i,j}$ for every mixed strategy leads to a point that lays in the

line connecting all the $R_{i,j}$. As a result, we have $\mathcal{P}_M(\text{supp}(\mathbf{x}_1), \text{supp}(\mathbf{x}_2)) \equiv \mathcal{P}_C(\text{supp}(\mathbf{x}_1), \text{supp}(\mathbf{x}_2))$. □

With can also extend this proof to the general n agents problem, given that each convex combination is strictly inside a n–dimensional polytope and the combination must be on a Pareto efficient edge, then the polytope must collapse to a line (1–dimensional polytope).

3.5 Smoothed complexity

Worstcase complexity, being too pessimistic, is often a bad indicator of the actual performance of an algorithm, and average–case complexity is difficult to determine. A new metric of complexity, called smoothed complexity, has been gaining interest in recent years, see [3]. It studies how the introduction of small perturbations affects the worst–case complexity. There might be several models of perturbations. The most common two perturbation models are the uniform perturbation and Gaussian perturbation.

In the case of stron Nash equilibrium finding problem, given a perturbation $\mathcal{D}_\sigma$ of magnitude σ, we have:

- uniform perturbation: every entry of U_i is subjected to an additive perturbation $[-\sigma, +\sigma]$ with uniform probability,
- Gaussian perturbation: every entry of U_i is subjected to an additive perturbation $[-z, +z]$ with probability $\frac{1}{\sigma\sqrt{2\pi}} e^{-|U_i(j,k)-z|^2/\sigma^2}$.

Denote by $\tilde{U}_i$ the perturbed utility matrix. The smoothed running time of an algorithm $\mathcal{A}$ given a perturbation $\mathcal{D}_\sigma$ is defined as follows:

$$\textsf{smoothed}\text{–}t_{\mathcal{A}} = \mathbb{E}_{\tilde{U}_1, \tilde{U}_2 \sim \mathcal{D}_\sigma}[t_{\mathcal{A}}(\tilde{U}_1, \tilde{U}_2) | U_1, U_2]$$

where $t_{\mathcal{A}}(\tilde{U}_1, \tilde{U}_2)$ is the running time for the game instance $(\tilde{U}_1, \tilde{U}_2)$. An algorithm has smoothed time complexity if for all $0 < \sigma < 1$ there are positive constants c, k_1, k_2 such that:

$$\textsf{smoothed}\text{–}t_{\mathcal{A}} = O(c \cdot m^{k_1} \cdot \sigma^{-k_2})$$

where m is the size of the game in terms of actions per agent. Basically, a problem is in smoothed–$\mathcal{P}$ if it admits a smoothed polynomial time algorithm. We can state the following theorem.

Theorem 3.5.1. *The problem of finding a strong Nash equilibrium is in smoothed–$\mathcal{P}$ class.*

Proof. Initially, we provide Algorithm 1. It is organized in three main parts.

In the first part (Steps 1–3), the algorithm searches for a pure–strategy strong Nash equilibrium by enumerating all the pure–strategy profiles and verifying whether each strategy profile is a strong Nash equilibrium. The verification is accomplished by checking whether or not Nash equilibrium constraints are satisfied (this can be done in polynomial time in m) and by checking whether or not the strategy profile is on the Pareto frontier (this can be done in polynomial time as shown in [12]). If a strong Nash equilibrium is found, the algorithm returns it, the algorithm continues otherwise. The maximum number of iterations in the first part of the algorithm is m^2.

In the second part (Steps 4–7), the algorithm verifies whether there there are strategy profile of supports $|\text{supp}(\mathbf{x}_1)| = |\text{supp}(\mathbf{x}_2)| = 2$ such that $\mathcal{P}_M(\text{supp}(\mathbf{x}_1), \text{supp}(\mathbf{x}_2)) = \mathcal{P}_C(\text{supp}(\mathbf{x}_1), \text{supp}(\mathbf{x}_2))$. This can be accomplished in polynomial time, i.e., $O(m^4)$, by checking whether there is a line connecting all entries of all the sub–bimatrix of size 2×2. In the affirmative case, the temporary variable temp is set true. Otherwise, temp is set false.

In the third part (Steps 8–13), if temp is false, the algorithm returns non–existence, given that, by Theorem 3.4.4, there is no mixed–strategy strong Nash equilibrium. Otherwise, the algorithm enumerate all the mixed–strategy profile and for each of them the algorithm verifies whether it is a strong Nash equilibrium as done in Steps 1–3. In the latter case, the algorithm can take exponential time.

Thus, the running time of Algorithm 1 is exponential if and only if it needs to enumerate an exponential number of supports during Steps 8–13. This happens only when $\mathcal{P}_M(\text{supp}(\mathbf{x}_1), \text{supp}(\mathbf{x}_2)) = \mathcal{P}_C(\text{supp}(\mathbf{x}_1), \text{supp}(\mathbf{x}_2))$ for some $\text{supp}(\mathbf{x}_1), \text{supp}(\mathbf{x}_2)$ with $|\text{supp}(\mathbf{x}_1)| = |\text{supp}(\mathbf{x}_2)| = 2$. However, given that the perturbations $\mathcal{D}_\sigma$ over all the entries of the utility matrices are independent and identically distributed, the probability that the perturbed entries are aligned as required by Theorem 3.4.4 to have strong Nash equilibria with $|\text{supp}(\mathbf{x}_1)| = |\text{supp}(\mathbf{x}_2)| > 1$ is zero. Therefore, the smoothed running time of Algorithm 1 is polynomial in m and independent of σ, for both uniform and Gaussian perturbations. As a result, finding a strong Nash equilibrium is a problem in smoothed–$\mathcal{P}$. □

The above result shows that, except for a space of the parameters with null measure, games admit only pure–strategy strong Nash equilibria and therefore that verifying the existence of a strong Nash equilibrium and finding them is computationally easy. Interestingly, the instability is due to the combination of Nash equilibrium constraints and Pareto efficiency con-

straints. Indeed, both the problem of finding a Nash equilibrium and the problem of finding Pareto efficiency are not sensible to perturbations.

Algorithm 1 findSNE

```
1: for all pure–strategy profiles do
2:    if the strategy profile is an SNE then
3:       return the strategy profile
4: temp ← false
5: for all strategy profile with |S_1| = |S_2| = 2 do
6:    if payoffs restricted to supports are aligned then
7:       temp ← true
8: if not temp then
9:    return non–existence
10: else
11:    for all non–pure–strategy profiles do
12:       if the strategy profile is an SNE then
13:          return the strategy profile
```

Chapter 4

Mathematical equilibrium constraints

In this chapter our intent is to provide a set of constraints for a strategy to be a strong Nash equilibrium, in a mathematical programming fashion. However, as showed in Section 2.2.3, the problem of finding Nash equilibria is equivalent to a nonlinear feasibility problem, whereas the problem of finding Pareto dominant stategies requires a multi–objective nonlinear optimization problem, as described in Section 2.2.1. The integration these two problems is not straightforward. In this chapter we provide different approximations through nonlinear feasibility programs that represent necessary *or* sufficient conditions.

At first we provide a mixed–integer nonlinear program to find Nash equilibria that are resilient to pure strategy multilateral deviations. We prove that, unlike in the case of Nash equilibrium, this condition is only necessary, and not sufficient, for a strategy to be a strong Nash equilibrium. Then we provide a nonlinear program to find a Nash equilibrium that satisfies the Karush–Kuhn–Tucker conditions. We prove that it is necessary, but not sufficient, for a strategy to be a strong Nash equilibrium. Moreover we provide a nonlinear program in which a strategy profile is forced to be Pareto efficient with respect to every possible coalition in correlated–strategies. We prove that this is sufficient, but not necessary, for a strategy to be a strong Nash equilibrium. In the end we characterize the relationships between the sets of solutions of these three formulations, the set of Nash equilibria, and the set of strong Nash equilibria.

4.1 Strong Nash equilibrium conditions

First of all we focus on the problem of deriving equilibrium constraints in a mathematical programming fashion for strong Nash equilibria.

Remark that, as stated by Lemma 2.2.12, a strategy profile $\mathbf{x}$ is a strong Nash equilibrium if and only if:

- it is a Nash equilibrium, that is, it satisfies constraints (2.6)–(2.9), and
- it is Pareto efficient for each non–singleton, non–empty coalition $C \in \mathcal{C}$. That is, for each C, it is an optimal solution to the nonlinear multi–objective optimization program in which the objective functions are the expected utilities of the agents that are members of C.

We already defined in Section 2.2.1 the Pareto efficiency problem through the nonlinear program (2.3), (2.4), (2.5), now we need to write the same problem for each coalition C. Given a strategy profile $\mathbf{x}$, we want to express the expected utilities of the agents that are members of C, for every conceivable non–singleton, non–empty coaltion. i.e. $EU_i(\mathbf{x})$, for all $i \in C$, for all $C \in \mathcal{C}$. Where $\mathcal{C}$ is a subset of $\mathcal{P}(N)$.

Remark that a_C is a profile of actions, i.e. a pure strategy profile, for each agent i, and conversely a_{-C} is a profile of actions for the agents that are not members of C. Moreover a (a_C, a_{-C}) is a pure strategy profile for the whole set of agents. By Definition 2.1.2, $U_i(a_C, a_{-C})$ is agent i's utility when the agents in C play the pure strategy a_C and the other agents play the pure strategy a_{-C}.

Remark also that, in the same way a strategy profile $\mathbf{x}$ can be partitioned in a strategy for the agents in C: $\mathbf{x}_C$, and a strategy for the other agents: $\mathbf{x}_{-C}$. Given a mixed–strategy $\mathbf{x}_C$, we denote by $x_i(a_C(i))$ the probability with which agent i, member of C, plays his action in the pure–strategy profile a_C, or equivalently, we denote by $a_C(i)$ the action of agent i in the pure–strategy a_C.

Therefore the expected utility for agent i can be expressed as the sum of the utility outcomes of every pure–strategy profile weighted by the probability of playing that strategy:

$$EU_i = \sum_{(a_C, a_{-C}) \in A} U_i(a_C, a_{-C}) \cdot \prod_{i \in C} x_i(a_C(i)) \cdot \prod_{j \notin C} x_j(a_{-C}(j))$$

Notice that the strategy of the agents outside the coalition C is not a variable, but it is rather a parameter of the problem. We replace the notation $x_j(a_{-C}(j))$ with $x_j(a_j)$ where a_j is the action of agent j in the given profile of actions a_{-C}.

Now we can write the Problem (2.3), (2.4), (2.5) for a generic coalition $C \in \mathcal{C}$ as:

$$\max_{\mathbf{x}_C} \left[\sum_{(a_C, a_{-C}) \in A} U_i(a_C, a_{-C}) \cdot \right.$$

$$\left. \cdot x_C(a_C) \cdot \prod_{j \notin C} x_j(a_j) : i \in C \right] \tag{4.1}$$

$$\text{subject to:} \quad x_C(a_C) = \prod_{i \in C} x_i(a_C(i)) \qquad \forall a_C \in A_C \tag{4.2}$$

$$x_i(a_i) \geq 0 \qquad \forall i \in C, a_i \in A_i \tag{4.3}$$

$$\sum_{a_i \in A_i} x_i(a_i) = 1 \qquad \forall i \in C \tag{4.4}$$

Here the objective function (4.1) is the vector of the expected utility functions of the agents i of the coalition C. Constraints (4.2) force the probability $x_C(a_C)$, i.e. the probability with which the coalition plays the pure–strategy profile a_C, to be equal to the product of the probabilities $x_i(a_C(i))$, with which each single agent i of C plays his action in a_C. Constraints (4.3) and (4.4) force $\mathbf{x}_i$ to be a well–defined strategy.

The satisfiability problem (2.6)–(2.9) and the multi–objective programs (4.1)–(4.4), one for each coalition C, constitute separate programs that must be satisfied together. However their integration is not straightforward. Consider, e.g., the class of bimatrix games. Only one coalition is possible: $C = \{1, 2\}$. If we solve program (4.1)–(4.4) for $C = \{1, 2\}$ with, as additional constraints, the Nash equilibrium constraints (2.6)–(2.9), we are searching for Nash equilibrium that is not Pareto dominated by other Nash equilibria. Instead, a strong Nash equilibrium is an Nash equilibrium that is Pareto efficient among all the strategy profiles, i.e. it is not Pareto dominated by any other strategy. This is not surprising, in fact, if we simply add new constraints to an optimization problem, we alter the search space and therefore we are going to find optima that are solutions to a different problem. Thus, in order to find a set of constraints for a strategy to be a strong Nash equilibrium, we need to translate, for each coalition C, program (4.1)–(4.4) into a feasibility problem that is satisfied by, and only by, all the optimal solutions of (4.1)–(4.4).

However the derivation of such necessary and sufficient equilibrium constraints is an open problem. It is elusive because program (4.1)–(4.4) is non–convex and therefore strong duality and complementary slackness con-

dition do not hold, unlike in the case of Nash equilibrium, where agents do not form coalitions.

In the next sections we derive necessary *or* sufficient conditions for a strategy to be a strong Nash equilibrium, using three different approaches, respectively.

4.2 Resilience to pure multilateral deviations

In this section we try to define Pareto efficiency requiring Pareto optimality of a solution with respect to only pure joint strategies of the coalition. That is, we require that no coalition can deviate from a generic strategy profile to a pure strategy profile that is strictly Pareto dominant for that coalition.

A motivation for this approach comes from what is known for Nash equilibria. Although the Nash equilibrium concept requires a strategy to be the best with respect to all the mixed strategies, it is sufficient to require that a strategy is best with respect to all pure strategy deviations.

Obviously, if we look at the Definition 2.2.11, resilience to pure multilateral deviations is a necessary condition for a strong Nash equilibrium. However, constraining a strategy to be Pareto optimal with respect to pure strategies of each coalition is not straightforward. A unilateral deviation is always possible, i.e., for each strategy, if there is an action that provides a better utility outcome, the agent will deviate. Instead, a multilateral deviation is possible if and only if all the members of the coalition can improve their utility reward by deviating. The need for activating and deactivating constraints related to a multilateral deviation pushes us to resort to mixed integer programming. Given a coalition C, we can formulate the constraints to force a Nash equilibrium to be resilient to pure multilateral deviations of C as a mixed integer nonlinear program:

$$r_{i,C}(a_C) \in \{0,1\} \qquad \begin{matrix}\forall a_C \in A_C,\\ i \in C\end{matrix} \tag{4.5}$$

$$v_i - \sum_{a_{-C} \in A_{-C}} U_i(a_C, a_{-C}) \cdot \prod_{j \in -C} x_j(a_j) \geq -M \cdot (|C| - \sum_{j \in C} r_{j,C}(a_C)) \qquad \begin{matrix}\forall i \in C,\\ a_C \in A_C\end{matrix} \tag{4.6}$$

$$v_i - \sum_{a_{-C} \in A_{-C}} U_i(a_C, a_{-C}) \cdot \prod_{j \in -C} x_j(a_j) \geq -M \cdot r_{i,C}(a_C) \qquad \begin{matrix}\forall i \in C,\\ a_C \in A_C\end{matrix} \tag{4.7}$$

where M is the largest payoff of all the agents. Constraints (4.5) force $r_{i,C}(a_C)$ to take on binary values. If the left hand side of (4.7) is negative, then there is a pure strategy profile (a_C, a_{-C}), where a_{-C} is fixed, that provides a better utility for agent i, in the coalition C. Therefore we have $r_{i,C}(a_C) = 1$, meaning that the deviation towards the pure strategy (a_C, a_{-C}) is active for agent i, member of coalition C. Constraints (4.6) force the left hand side to be positive if the variables $r_{i,C}(a_C)$ of all the members of the coalitions are set to one. That is, if playing (a_C, a_{-C}), where as usual a_{-C} is fixed, is the best for all the members of the coalitions, then this multilateral deviation is active and the utility of each member of the coalition must be at least the utility given by playing (a_C, a_{-C}), that is $v_i \geq EU_i(a_C, a_{-C})$, for every agent i in C.

Formulation 4.2.1. The problem of finding a Nash equilibrium that is resilient to pure multilateral deviations ($NEPMD$) can be formulated as:

- Constraints (2.6)–(2.9)
- Constraints (4.5)–(4.7) for all $C \in \mathcal{C}$

If there are only two agents, this set of constraints constitutes a mixed integer *linear* program, given that the Nash equilibrium constraints can be expressed in mixed integer linear fashion [24].

We can state the following theorem.

Theorem 4.2.2. *The constraints 2.6–2.9, 4.5–4.7, ($NEPMD$) are necessary conditions for a strategy profile to be a strong Nash equilibrium.*

Proof. By Lemma 2.2.12 a strategy profile to be a strong Nash equilibrium has to be a Nash equilibrium and by Definition 2.2.11 a strong Nash equilibrium is resilient to multilateral deviations. If a strategy is resilient to multilateral deviations it immediately follows that it is also resilient to pure multilateral deviations (i.e. a propre subset of all the possible multilateral deviations). Hence if a strategy is a strong Nash equilibrium the problem (2.6)–(2.9), (4.5)–(4.7) for all $C \in \mathcal{C}$, is feasible. □

However, unlike in the case of Nash equilibrium, these conditions are not sufficient, in fact:

Theorem 4.2.3. *Resilience to pure multilateral deviations is not sufficient for a Nash equilibrium to be a strong Nash equilibrium.*

Proof. Consider the two–agents game described in Table 4.1. In this case there are three Nash equilibria:

- $\mathrm{NE}_1 = (\mathsf{a}_3, \mathsf{a}_6)$,
- $\mathrm{NE}_2 = (\frac{1}{2}\mathsf{a}_1 + \frac{1}{2}\mathsf{a}_2, \frac{1}{2}\mathsf{a}_4 + \frac{1}{2}\mathsf{a}_5)$
- $\mathrm{NE}_3 = (\frac{1}{7}\mathsf{a}_1 + \frac{1}{7}\mathsf{a}_2 + \frac{5}{7}\mathsf{a}_3, \frac{1}{7}\mathsf{a}_4 + \frac{1}{7}\mathsf{a}_5 + \frac{5}{7}\mathsf{a}_6)$.

The first one is a pure strategy, and the other two are mixed strategies. Focus on NE_1: for both agents the expected utility is 1 and there is no outcome achievable by pure strategy multilateral deviations that provides both agents a utility strictly greater than 1. For instance, the pure strategy $(\mathsf{a}_1, \mathsf{a}_4)$ is better for agent 1 than NE_1, but this does not hold for agent 2. With $(\mathsf{a}_2, \mathsf{a}_4)$ we have the reverse. However, NE_1 is not weakly Pareto efficient, as shown by the Pareto frontier in Figure 4.1. Indeed, the mixed–strategy NE_2 strictly Pareto dominates NE_1, providing an expected utility of 1.25 for both agents. The former, being a Nash equilibrium that lies on the Pareto frontier, is a strong Nash equilibrium. □

		agent 2		
		a_4	a_5	a_6
agent 1	a_1	5,0	0,5	0,0
	a_2	0,5	5,0	0,0
	a_3	0,0	0,0	1,1

Table 4.1: Example of a game with a Nash equilibrium which is resilient to pure–strategies multilateral deviations but is not a strong Nash equilibrium, being Pareto dominated.

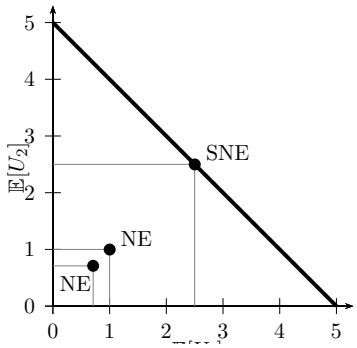

Figure 4.1: The Pareto frontier of the game described in Table 4.1.

The conditions (2.6)–(2.9), (4.5)–(4.7) for all $C \in \mathcal{C}$, are stronger than the Nash equilibrium conditions, e.g, see the prisoner dilemma, described in Table 2.2, therefore the following proposition holds.

Proposition 4.2.4. *There are Nash equilibria that do not satisfy constraints (4.5)–(4.7).*

4.3 Karush–Kuhn–Tucker Conditions

In this section we present our second formulation approach, which is based on Karush–Kuhn–Tucker conditions. In Section 2.3.1 we defined the Karush-Kuhn-Tucker conditions as first order necessary conditions for a solution in nonlinear programming to be optimal. As in Chapter 3 the idea is to use these conditions to convert the Pareto optimality problem (2.3), (2.4), (2.5), into a feasibility problem (2.13)–(2.16) which is a set of first order necessary conditions for local Pareto efficiency.

In Secrion 3.1.1 we already showed how it is possible to derive the Karush–Kuhn–Tucker conditions for a bimatrix game, now we can map these conditions to the case of Pareto efficiency for a single coalition C as follows:

- f_i: is agent i's expected utility;
- g_j: is a constraint of the form $x_w(a_w) \geq 0$;
- h_k: is a constraint of the form $\sum_{a_i \in A_i} x_i(a_i) - 1 = 0$.

Given a coalition C, we obtain the following conditions:

$$\sum_{i \in C} \lambda_{i,C} \cdot \sum_{\substack{a \in A: \\ a_w = \overline{a}_w}} U_i(a) \cdot \prod_{\substack{j \in C: \\ j \neq w}} x_j(a_j) \cdot$$

$$\cdot \prod_{j \notin C} x_j(a_j) + \mu_C(\overline{a}_w) + \nu_{w,C} = 0 \qquad \begin{matrix} \forall w \in C, \\ \overline{a}_w \in A_w \end{matrix} \tag{4.8}$$

$$\mu_C(a_i) \cdot x_i(a_i) = 0 \qquad \forall i \in C, a_i \in A_i \tag{4.9}$$

$$\lambda_{i,C} \geq 0 \qquad \forall i \in C \tag{4.10}$$

$$\mu_C(a_i) \geq 0 \qquad \forall i \in C, a_i \in A_i \tag{4.11}$$

$$\sum_{i \in C} \lambda_{i,C} = 1 \tag{4.12}$$

Notice that, when using a modelling language, such as AMPL [10], the irreflexive order relation $<$ is not supported, hence, in this, formulation we approximate constraint (2.15) through constraint (4.10) and (4.12).

Now, we can leverage the above results, producing a nonlinear mathematical program for strong Nash equilibrium.

Formulation 4.3.1. The problem of finding a Nash equilibrium such that the Karush–Kuhn–Tucker conditions are satisfiable ($NEKKT$) can be formulated as:

- Constraints (2.6)–(2.9),
- Constraints (4.8)–(4.12) $\forall C \in \mathcal{C}$

We can state the following theorem.

Theorem 4.3.2. *The satisfiability of constraints (2.6)–(2.9), (4.8)–(4.12) $\forall C \in \mathcal{C}$, ($NEKKT$) is a necessary condition for a strategy profile to be a strong Nash equilibrium.*

Proof. By Lemma 2.2.12 a strategy profile to be a strong Nash equilibrium has to be a Nash equilibrium and has to be Pareto efficient for every possible coalition $C \in \mathcal{C}$. Moreover a strategy is a Nash equilibrium if and only if satisfies the set of constraints (2.6)–(2.9). On the other hand for every coalition C a strategy that is Pareto efficient satisfies the set of constraints (4.8)–(4.12), as the Karush-Kuhn-Tucker conditions are, by definition, necessary conditions. Therefore if a strategy profile is a strong Nash equilibrium it satisfies constraints (2.6)–(2.9), (4.8)–(4.12) $\forall C \in \mathcal{C}$. □

However, we can show that the above conditions are not sufficient.

Theorem 4.3.3. *The satisfiability of constraints (2.6)–(2.9), (4.8)–(4.12) $\forall C \in \mathcal{C}$, ($NEKKT$) is not a sufficient condition for a strategy profile to be a strong Nash equilibrium, nor for Nash equilibria to be locally Pareto efficient.*

Proof. Consider the game in Table 4.2. The game admits a mixed strategy Nash equilibrium $(\frac{1}{2}a_1 + \frac{1}{2}a_2, \frac{1}{2}a_5 + \frac{1}{2}a_6)$ that gives each of the two agents utility 1. This Nash equilibrium is Pareto dominated by, e.g., the mixed strategy $(\frac{1}{2}a_3 + \frac{1}{2}a_4, \frac{1}{2}a_7 + \frac{1}{2}a_8)$ that gives each agent utility $\frac{9}{4}$. Moreover the previous Nash equilibrium satisfies the Karush-Kuhn-Tucker conditions, which, in a bimatrix game, can be expressed through constraints (3.6)–(3.11), for all feasible λ_1, λ_2 with $\mu_1(a_3) = \mu_1(a_4) = \mu_2(a_7) = \mu_2(a_8) = 6$ and $\nu_1 = \nu_2 = 1$. □

However, we can also show that not every Nash equilibrium satisfies the Karush–Kuhn–Tucker conditions.

Proposition 4.3.4. *There are Nash equilibria that do not satisfy the set of constraints (4.8)–(4.12).*

		agent 2			
		a_5	a_6	a_7	a_8
agent 1	a_1	2,0	0,2	-5,-5	-5,-5
	a_2	0,2	2,0	-5,-5	-5,-5
	a_3	-5,-5	-5,-5	5,0	0,0
	a_4	-5,-5	-5,-5	0,0	0,5

Table 4.2: Example of a game with a Nash equilibrium that satisfies Karush–Kuhn–Tucker conditions even though it is Pareto dominated.

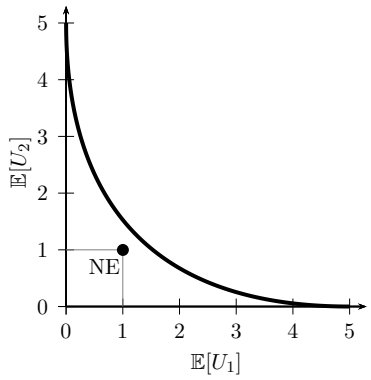

Figure 4.2: The Pareto frontier of the game described in Table 4.2.

Proof. Consider the prisoners dilemma, described in Table 2.2. We show that conditions (3.6)–(3.11) are not satisfied considering the strategy $(\mathsf{a}_2, \mathsf{a}_4)$, which is a Nash equilibrium. The Karush–Kuhn–Tucker conditions at $x_1(\mathsf{a}_2) = x_2(\mathsf{a}_4) = 1$ are:

$$\begin{aligned}
&\mu_1(\mathsf{a}_2) = 0\\
&\mu_2(\mathsf{a}_4) = 0\\
&5\lambda_2 + \mu_1(\mathsf{a}_1) = \nu_1\\
&\lambda_1 + \lambda_2 = \nu_1\\
&5\lambda_1 + \mu_2(\mathsf{a}_3) = \nu_2\\
&\lambda_1 + \lambda_2 = \nu_2
\end{aligned}$$

By straightforward mathematics, we obtain:

$$3\lambda_1 + 3\lambda_2 = -\mu_1(\mathsf{a}_1) - \mu_2(\mathsf{a}_3)$$

Given that $\lambda_i, \mu_i(a_i) \geq 0$ and $\sum_{i \in N} \lambda = 1$, the above equality is unsatisfiable and therefore Karush–Kuhn–Tucker conditions are not satisfied. □

4.4 Correlated Pareto frontier

In this section we present our third formulation approach. In order to provide sufficient conditions, we relax the constraints of program (4.1)–(4.4), allowing the agents belonging to coalition C to play correlated strategies. By Proposition 2.1.8 when correlated strategies are allowed the Pareto frontier becomes a convex combination of known points, hence we can write a *linear* program to find Pareto efficient points.

Remark that correlated strategies include all the mixed strategies. Thus, if a strategy profile of the members of coalition C is the best with respect to all their correlated strategies, then it is the best also with respect to all the mixed strategies. However, requiring Pareto optimality with respect to correlated strategies, we may discard solutions that are optimal when correlated strategies are not allowed.

At first, we reformulate program (4.1)–(4.4) when the members of coalition C can play correlated strategies. We already defined $x_C(a_C)$ in Equation (4.2) as the probability with which the coalition plays the pure–strategy profile a_C. Hence the vector $\mathbf{x}_C$ represents the probability distribution over the set of pure–strategy profiles A_C, i.e. it is the correlated strategy profile of the agents in the coalition C. More precisely the Equation (4.2) binds correlated strategy to mixed strategy, therefore this constraint limits the search space to the correlated strategies that are indeed mixed strategies. As a correlated strategy is a probability distribution each element $x_C(a_C)$ has to be greater than, or equal to zero and all the elements in $\mathbf{x}_C$ must sum to one. We can state these properties with the following constraints.

$$x_C(a) \geq 0 \qquad \forall a \in A_C \tag{4.13}$$

$$\sum_{a \in A_C} x_C(a) = 1 \tag{4.14}$$

Notice that the optimization problem with objective function (4.1) and constraints (4.13), (4.14) is linear in $\mathbf{x}_C$, as the $x_j(a_j)$ are parameters of the problem. If a solution $\mathbf{x}_C$ is optimal for program (4.1), (4.13), (4.14), then there is a vector of multipliers $\lambda_i \geq 0$ in which at least one multiplier is strictly positive such that $\mathbf{x}_C$ is an optimal solution of the following problem:

$$\max_{\mathbf{x}_C} \sum_{i \in C} \lambda_{i,C} \cdot \sum_{(a_C, a_{-C}) \in A} U_i(a_C, a_{-C}) \cdot x_C(a_C) \cdot \prod_{j \notin C} x_j(a_j) \tag{4.15}$$

$$\text{Subject to: } \lambda_{i,C} \geq 0, \ \forall i \in C, \tag{4.16}$$

$$\sum_{i \in C} \lambda_{i,C} = 1, \tag{4.17}$$

$$\text{constraint (4.13)}$$

$$\text{constraint (4.14)}$$

We can now derive the dual problem s:

$$\min_{v_C} v_C \tag{4.18}$$

$$\text{subject to: } v_C \geq \sum_{i \in C} \lambda_{i,C} \cdot \sum_{a_{-C} \in A_{-C}} U_i(a_C, a_{-C}) \cdot \cdot \prod_{j \notin C} x_j(a_j) \forall a_c \in A_c \tag{4.19}$$

where v_C is the dual variable of $\mathbf{x}_C$. Given that the primal problem is convex, strong duality holds and we can apply the complementary slackness theorem [4], obtaining the following feasibility problem:

$$x_C(a_c) \cdot \left(v_C - \sum_{i \in C} \lambda_{i,C} \cdot \sum_{a_{-C} \in A_{-C}} U_i(a_C, a_{-C}) \cdot \cdot \prod_{j \notin C} x_j(a_j) \right) = 0, \ \forall a_c \in A_c \tag{4.20}$$

$$\text{constraint (4.13)}$$

$$\text{constraint (4.14)}$$

$$\text{constraint (4.19)}$$

The above constraints are sufficient conditions for a correlated–strategy profile $\mathbf{x}_C$ to be Pareto efficient, once the strategies of agents outside C are fixed. Remark that this problem is linear in $\mathbf{x}_C$. However if we want to include the Nash equilibrium conditions, we can not limit our analysis to correlated strategies, and therefore the problem is still nonlinear. Now, we can use the above results to produce a nonlinear program for strong Nash equilibrium.

Formulation 4.4.1. The problem of finding a Nash equilibrium that is Pareto efficient when correlated strategies are allowed, with respect to every conceivable coalition ($corrSNE$) can be formulated as:

- Constraints (2.6)–(2.9),
- Constraint (4.2), (4.16), (4.17), (4.19), (4.20) $\forall C \in \mathcal{C}$.

where constraints (2.6)–(2.9) assure that $\mathbf{x}$ is a Nash equilibrium; constraints (4.19), (4.20) $\forall C \in \mathcal{C}$ and constraints (4.16), (4.17) assure that, for every C, there are some well defined multipliers $\lambda_{i,C}$ such that $\mathbf{x}_C$ is optimal among all the correlated strategies and therefore $\mathbf{x}_C$ is Pareto efficient; constraint (4.2) guarantees that there exists a mixed strategy $\mathbf{x}$ matching the correlated strategy $\mathbf{x}_C$. We can now state the following theorem.

Theorem 4.4.2. *The constraints (2.6)–(2.9), (4.2), (4.16), (4.17), (4.19), (4.20) $\forall C \in$, (corrSNE) are sufficient conditions for a strategy profile to be a strong Nash equilibrium.*

Proof. It is sufficient to notice that the correlated–Pareto frontier dominates the Pareto frontier. □

However, the above conditions are not necessary:

Theorem 4.4.3. *The constraints (2.6)–(2.9), (4.2), (4.16), (4.17), (4.19), (4.20) $\forall C \in$, (corrSNE) are not necessary conditions for a strategy profile to be a strong Nash equilibrium.*

Proof. Consider the game described in Table 4.3, where $\rho \in [0, 2)$. The game has one strong Nash equilibrium, i.e., $\text{SNE} = (\mathsf{a}_3, \mathsf{a}_6)$, but this equilibrium does not satisfy the above constraints. Indeed, the agents' utilities at $(\mathsf{a}_3, \mathsf{a}_6)$, i.e., $(2, 2)$, are not on the correlated–strategy Pareto frontier, i.e., the dashed line connecting (5,0) to (0,5). Thus, the above nonlinear mathematical program is infeasible. □

		agent 2		
		a_4	a_5	a_6
agent 1	a_1	5,0	0,0	0,ρ
	a_2	0,0	0,5	0,ρ
	a_3	ρ,0	ρ,0	2,2

Table 4.3: Example of a game with a Nash equilibrium which does not lie on the correlated–strategy Pareto frontier.

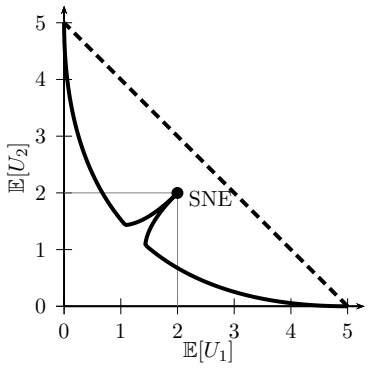

Figure 4.3: The Pareto frontier of the game described in Table 4.3.

4.5 Relationships between the solutions

We now study the relationships between the solutions found by the formulations provided in the previous sections. Given a strategic form game, call *NEPMD* the set of strategies that are Nash equilibria and that are resilient to pure multilateral deviations. Let *NEKKT* be the set of strategies that are Nash equilibria and that satisfy the Karush–Kuhn–Tucker conditions. We can state the following proposition.

Proposition 4.5.1. *The following relationships hold:*

- $NEKKT \not\subseteq NEPMD$,
- $NEPMD \not\subseteq NEKKT$.

Proof. We prove this by counterexamples. Consider the game described by Table 4.4. It is easy to show that the Nash equilibrium $(\frac{1}{3}\mathsf{a}_1 + \frac{2}{3}\mathsf{a}_2, \frac{3}{4}\mathsf{a}_3 + \frac{1}{4}\mathsf{a}_4)$ is resilient to pure multilateral deviations, hence it is in the set *NEPMD*. However it does not satisfy the Karush–Kuhn–Tucker conditions and therefore it does not belong to *NEKKT*. This shows that there are elements in the set $NEPMD \setminus NEKKT$, obviously elements of this set are not strong Nash equilibria. It follows that *NEKKT* is not a subset of *NEPMD*.

		agent 2	
		a_3	a_4
agent 1	a_1	1,0	0,2
	a_2	0,1	3,0

Table 4.4: Example of a game with a Nash equilibrium which is resilient to pure–strategies multilateral deviations but does not satisfy the Karush–Kuhn–Tucker conditions.

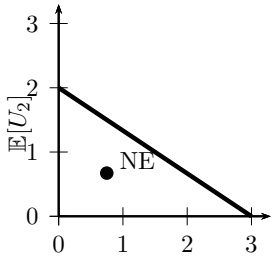

Figure 4.4: The Pareto frontier of the game described in Table 4.4.

Consider now the game described in Table 4.5. By trivial mathematics it is possible to verify that the Nash equilibrium $(\mathsf{a}_3, \mathsf{a}_6)$ satisfies the Karush–Kuhn–Tucker conditions, thus it belongs to *NEKKT*. However it is not resilient to pure strategies multilateral deviations as both $(\mathsf{a}_1, \mathsf{a}_4)$ and $(\mathsf{a}_2, \mathsf{a}_5)$ provide a better utility for both agents, and so it does not belong to *NEPMD*. This shows that there are elements in the set $NEKKT \setminus NEPMD$, and, like in the previous case, these elements are not strong Nash equilibria. Therefore *NEPMD* is not a subset of *NEKKT*. □

		agent 2		
		a_4	a_5	a_6
agent 1	a_1	5,2	0,0	0,0
	a_2	0,0	2,5	0,0
	a_3	0,0	0,0	1,1

Table 4.5: Example of a game with a Nash equilibrium which satisfies the Karush–Kuhn–Tucker conditions but is not resilient to pure–strategies multilateral deviations.

Now, given a strategic form game, call *SNE* the set of all the strategies that are strong Nash equilibria, and let *corrSNE* the set of strategies that are both Nash equilibria and Pareto efficient, with respect to the correlated strategies. We can prove the following proposition.

Proposition 4.5.2. *The following relationship holds:*

- $SNE \subset (NEKKT \cap NEPMD)$,
- $corrSNE \subset SNE$.

Proof. As for the first relation, the subset relation follows by Theorem 4.3.2 and Theorem 4.2.2. In fact, as both the satisfiability Karush–Kuhn–Tucker conditions

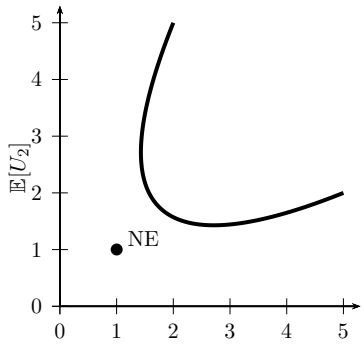

Figure 4.5: The Pareto frontier of the game described in Table 4.5.

and the resilience to pure multilateral deviations towards pure strategies are necessary conditions for a strategy to be a strong Nash equilibrium, any strategy that belongs to *SNE* also belongs to *NEKKT* and to *NEPMD*. Furthermore *SNE* is a proper subset of (*NEKKT* ∩ *NEPMD*). Consider the game described in Table 4.2, the strategy NE = $(\frac{1}{2}a_1 + \frac{1}{2}a_2, \frac{1}{2}a_5 + \frac{1}{2}a_6)$ is a Nash equilibrium. Moreover the strategy NE satisfies the Karush–Kuhn–Tucker conditions and is also resilient to multilateral deviations to pure strategies. Hence NE $\in NEPMD$ and NE $\in NEKKT$, but it is not an strong Nash equilibrium, as showed in Figure 4.2, therefore NE $\notin SNE$.

The second relation is an immediate consequence of Theorem 4.4.2 and Theorem 4.4.3. □

We can summarize the relationships between the solutions as:

$$
\begin{aligned}
corrSNE \subset SNE \subset (NEKKT \cap NEPMD) \\
(NEKKT \cap NEPMD) \subset NEPMD \subset NE \\
(NEKKT \cap NEPMD) \subset NEKKT \subset NE \\
NEPMD \nsubseteq NEKKT \\
NEKKT \nsubseteq NEPMD
\end{aligned}
$$

The (strict) inclusion relation induces a partial order among the defined sets, which can be represented through a Hasse diagram, as shown in Figure 4.6.

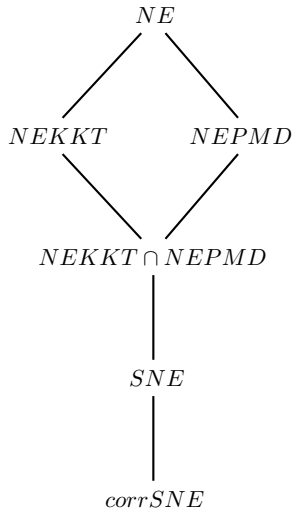

Figure 4.6: The Hasse diagram representing the partial order induced by the (strict) inclusion relation among the sets $corrSNE$, SNE, $NEKKT \cap NEPMD$, $NEKKT$, $NEPMD$ *and* NE.

Chapter 5

Iterative Algorithm for Strong Nash equilibrium

In this chapter we present an algorithm for finding a strong Nash equilibrium, able to handle mixed strategies, which is an extension of a tree search algorithm [12], developed for the same purpose. The prior algorithm works only in the case of two–agents game. Our aim is to design a more general algorithm which can handle games with three or more agents.

The basic idea is to use a nonlinear solver to find a strategy which is a solution to one of the problems presented in Chapter 4, then check whether this solution is Pareto efficient, with respect to every conceivable non–singleton coalition. If this is the case then the strategy is a strong Nash equilibrium, otherwise the process is repeated, limited to a subpspace of the problem, in a branch–and–bound fashion. However the use of nonlinear programming introduces incompleteness issues, due to the algorithm used by the nonlinear solver.

In the first section of this chapter we describe how the algorithm works. In particular we can implement different algorithms using different oracles, i.e. functions that provide strategies that are strong Nash candidate, one for each set of necessary conditions (for a strategy to be a strong Nash equilibrium) defined in Chapter 4. In the second section we discuss on how the incompleteness of the nonlinear solver affects our algorithm. Finally, in the last part we describe how we designed our tests and we discuss the results of the experiments.

5.1 Iterative strong Nash algorithm

In our previous analysis we could not derive a finite set of necessary and sufficient constraints for a strategy to be a strong Nash equilibrium, and thus we can not demand the computation of equilibria to a (nonlinear) mathematical programming solver. Therefore we propose an algorithm that iterates between the computation of a startegy that is a strong Nash candidate and its verification, which is logically an extension of the algorithm presented in [12].

The algorithm we propose is essentially a *spatial branch–and–bound* algorithm, which uses an oracle to produce a candidate solution and then verifies whether or not it is a strong Nash equilibrium. If not, a dominant strategy exists, and the algorithm computes a new candidate solution in a subspace induced by the dominant strategy. This process repeats until a strong Nash equilibrium is found or it is proven that none exists. The oracle essentially consists in the invocation of a nonlinear solver on one of the sets of necessary conditions for a strategy to be a strong Nash equilibrium.

5.1.1 Operative principles

The existing algorithm, described in [12], iterates between the computation of a Nash equilibrium, computed through a MIP Nash algorithm, and the verification of its Pareto efficiency. However, MIP Nash is based on integer linear programming, and with more than two agents the Nash equilibrium-finding program becomes nonlinear. Hence if we want to use mathematical programming we have no other choice than using nonlinear models. For the oracle, we adopt our formulations for Nash equilibrium, Formulation 2.2.8, Nash equilibrium resilient to pure multilateral deviations, Formulation 4.2.1, Nash equilibrium satisfying Karush–Kuhn–Tucker conditions, Formulation 4.3.1, and Nash equilibrium satisfying both Formulation 4.2.1 and Formulation 4.3.1. As we proved, these are all sets of necessary, but not sufficient, conditions for a strategy to be a strong Nash equilibrium.

Using a proper subset of the Nash equilibria to generate the solution candidates, we also aim to reduce the tree search depth. In fact, assume that candidate solutions are sampled with uniform probability distribution, let:

- $P_N = \frac{|SNE|}{|NE|}$ be the probability that a candidate from NE is a strong Nash equilibrium,
- $P_K = \frac{|SNE|}{|NEKKT|}$ be the probability that a candidate from $NEKKT$ is a strong Nash equilibrium,
- $P_M = \frac{|SNE|}{|NEPMD|}$ be the probability that a candidate from $NEPMD$ is a strong Nash equilibrium,
- $P_{MK} = \frac{|SNE|}{|NEPMD|}$ be the probability that a candidate from $NEPMD \cap NEKKT$ is a strong Nash equilibrium;

We showed in Section 4.5 the relationships among these sets, in particular we have that $NEKKT$ and $NEPMD$ are, in general, proper subsets of NE, therefore $|NE| > |NEKKT|$ and $|NE| > |NEPMD|$. We also showed that $NEPMD \cap NEKKT$ is a proper subset of both $NEPMD$ and $NEKKT$. This implies that the following relations hold:

- $P_N < P_K < P_{MK}$,
- $P_N < P_M < P_{MK}$,

5.1.2 Algorithm details

The algorithm is reported in Algorithm 2 and works as follows. The parameter space is the n–dimensional space where candidate solutions have to be searched for and it is initialized with the whole agents' utility space. Notice that the agents' utility space is can be defined as $s = [U_1^{min}, U_1^{max}] \times \cdots \times [U_n^{min}, U_n^{max}]$. At first a solution candidate is generated by the oracle within the given space. As the oracle uses a set of necessary conditions we assume that if a candidate solution can not be generated, then a solution does not exists in the given search space. However, if the oracle fails to find any candidate solution, as the nonlinear mathematical program of the oracle is solved with an incomplete algorithm, a solution may indeed exists, thus the overall algorithm is incomplete. We will further discuss these issues in the following section.

Algorithm 2 iteratedFindSNE(*space*)

Require: The search *space*

1: $\mathbf{x} \leftarrow$ callOracle(space)
2: **if** $\mathbf{x} = \varnothing$ **then**
3: **return** $\varnothing$
4: $isSNE \leftarrow$ true
5: **for all** $C \in \mathcal{C}$ **do**
6: $\mathbf{x}' =$ findParetoDominantStrategy$(\mathbf{x}, C)$
7: **if** $\mathbf{x}' \neq \varnothing$ **then**
8: $isSNE \leftarrow$ false
9: **for all** $i \in N$ **do**
10: $\mathbf{x} \leftarrow$ iteratedFindSNE(subSpace$(space, \mathbf{x}', i)$
11: **if** $\mathbf{x} \neq \varnothing$ **then**
12: **return** $\mathbf{x}$
13: **if** $isSNE$ **then**
14: **return** $\mathbf{x}$
15: **return** $\varnothing$

Once a candidate solution is found, we check whether or not it is a strong Nash equilibrium. The function isParetoDominant$(\mathbf{x}, C)$ checks if the strategy $\mathbf{x}$ is Pareto dominated with respect to the coalition C. Given a coalition C, and a strategy $\mathbf{x}$, the strategies of the players outside C, $\bar{\mathbf{x}}_{-C}$ are fixed, then we look for a strategy $\mathbf{x}' = (\mathbf{x}'_C, \bar{\mathbf{x}}_{-C})$ that Pareto dominates $\mathbf{x}$. If such a strategy does not exists then $\mathbf{x}$ is Pareto efficient with respect to the coalition C and the functions returns the value $\varnothing$. Conversely, the candidate is not a strong Nash equilibrium and the functions returns a $\mathbf{x}'$, which dominates $\mathbf{x}$.

Given a solution candidate $\mathbf{x}$ we have to check if it is Pareto efficiency with respect to every non–singleton coalition $C \in \mathcal{C}$. If this is the case then it is a strong Nash equilibrium and, after the for cycle, at Steps 5–12, the value of $isSNE$ is

true, therefore, at Step 14, the algorithm terminates and returns the strong Nash equilibrium $\mathbf{x}$. Otherwise, at some iteration of the cycle at Steps 5–12, a dominant strategy $\mathbf{x}'$ will be found, the variable $isSNE$ is set to false and we proceed with the branch step.

The branch–and–bound step consists in a partition of the search space. Given an agent i, if the candidate solution $\mathbf{x}$ is not a strong Nash equilibrium, then there exists a dominant (with respect to a coalition $C \in \mathcal{C}$) strategy $\mathbf{x}'$, therefore a subspace of the search space is generated with $EU_i(\mathbf{x}')$ as a lower bound for the utility of the i th agent. If we denote $\bar{U}_i = EU_i(\mathbf{x}')$, the subspace induced by agent i is defined by $s' = S_1 \times \cdots \times S_n$, where S_k is the utility range for the k–th agent. Therefore $S_k = [\bar{U}_i, U_i^{max}]$ if the k–th agent is the agent–i, $S_k = [U_j^{min}, U_j^{max}]$ for each agent–j different than agent–i. The function subSpace$(space, \mathbf{x}', i)$ generates the subspace induced by the strategy $\mathbf{x}'$, which dominates $\mathbf{x}$, and agent i, in the current space.

This partition step allows to exclude the previous candidate solution $\mathbf{x}$ form the search space. As for each agent a different subspace is induced, we have that the branching factor is equal to the number of agents for every coalition $C \in \mathcal{C}$, where a dominant strategy can be found. The worst case is when a solution candidate is Pareto dominated in every coalition, and therefore the branching factor is $b = n \cdot (2^n - n - 1)$.

Once the search subspace is defined the algorithm recursively calls itself until a solution is found or the empty set is returned. At every branch, if a solution is found then the current iteration terminates and returns this solution. Otherwise, if the search in every subspace returns no solution then the execution reaches Step 15 and the empty set is returned, as it is assumed that, in the current space, the oracle is unable to generate a candidate solution that is a strong Nash equilibrium.

The algorithm behaves as a depth–first tree search algorithm. In fact we can consider every node of the tree as a different search space, and the i–th child of a node is the subspace of the parent, induced by the utility of the i–th agent. The root node is the whole agents' utility space.

5.1.3 Implementation

We implemented the algorithm using Java programming language. For the oracle function we used AMPL [10] as the modeling language to provide our formulation and SNOPT (Sparse Nonlinear OPTimizer) [27] to solve them. The algorithm is implemented for three–agents games, however, using the general formulations provided in the previous chapters, it is possible to generalize it to the m agents case. In Appendix A the AMPL translations for the three–players case of Formulation 4.2.1, Formulation 4.3.1 and Formulation 4.4.1 are reported.

5.2 Nonlinear solver issues

As stated in the previous section, the oracle function provides a solution for a nonlinear program using SNOPT, the main issue is that its core, a sparse sequential

quadratic programming algorithm, is not complete. Therefore it may be unable to find a solution for the feasibility problems, that represent the strong Nash necessary conditions, even though a solution actually exists. Remark that, differently from the case of Nash equilibrium, a strong Nash equilibrium may not exists, therefore we need a stop condition for a spatial search algorithm. Our algorithm relies on the necessity hypothesis to check whether the equilibrium exists or not, i.e. in a given search space, if the necessary conditions can not be satisfied, then it is assumed that a strong Nash equilibrium does not exists. However these conditions may be turn out not satisfiable due to the incompleteness of the nonlinear solver, rather than because a solution does not actually exists. Therefore an algorithm that relies on this kind of solver to find solution candidates is incomplete. We can state the following theorem.

Theorem 5.2.1. *The Algorithm 2 is not complete.*

We use the same nonlinear solver to check wether a strategy is Pareto dominated, at least for the grand coalition in three–agents games. We check if there exists a strategy, different from the given one, that provides a better utility outcome. If such a strategy does not exists, then we consider the given strategy Pareto efficient. Again the solver may be unable to find a dominant strategy because of its incompleteness, therefore we might consider a strategy Pareto efficient even when it is actually dominated. As if a solution candidate is found to be Pareto efficient with respect to every conceivable coalition, it is returned as a solution, it turns out that our algorithm may find solutions that are not strong Nash equilibria. We can state the following theorem.

Theorem 5.2.2. *The Algorithm 2 is not sound.*

However, using an incomplete nonlinear solver we can still design a sound algorithm, that we can use for control purposes, using a set of sufficient conditions for a strategy to be a strong Nash equilibrium. In fact, if a strategy satisfies sufficient conditions, the correctness of the nonlinear solver guarantees that this strategy is a strong Nash equilibrium. We can state the following theorem.

Theorem 5.2.3. *The algorithm that finds a strong Nash equilibrium by solving the problem defined by Formulation 4.4.1 (corrSNE), using a nonlinear solver, is sound but not complete.*

Notice that, by Theorem 4.4.3, it is clear that such an algorithm would not be complete even with a complete solver.

A typical fix for the problem of the incompleteness of some nonlinear optimization algorithms is using random restarts. Thus if there is a set of start points which leads to local optima or, in the worst case, to singular points, setting many different random start points can give a chance to avoid such undesired situations. In our algorithm we implemented this technique as follows: when invoking the oracle function a maximum number of random restarts has to be provided. If the first execution of the oracle, with a random start point, that is a random initial strategy

profile for each agent, fails to find a strategy that satisfies the given set of necessary condition, then the oracle is executed again with a different initial strategy. This process repeats until a candidate solution is found or the given number of maximum. In the worst case, i.e. there are no strategies satisfying the given set of conditions, the oracle function is executed a number of times that matches the given number of maximum random restarts.

5.3 Experimental results

As the algorithm is not sound nor complete we are interested in evaluating its accuracy, and how it is affected by the size of the game and the number of maximum restarts. Moreover when a solution is found we interested in details on the algorithm performances such as the tree depth and the average time. The main issue is that to test the accuracy it is necessary to know in advance whether a given game has a strong Nash equilibrium or not, hence we have to define proper test sets.

5.3.1 Test settings and objectives

First of all we focus on the total problem of finding Nash equilibria. We use SNOPT to solve the nonlinear program defined by in (2.6)–(2.9), with uniform random restarts over the strategy space, in order to find a Nash equilibrium. The accuracy is measured as the number of games where the problem is solved, over the total number games in the test set. As the set of conditions (2.6)–(2.9) is a subset of every formulation in Chapter 4, it is meaningful to check if the nonlinear solver is able to find a strategy that satisfies this problem with sufficient reliability. We expect the reliability to be an increasing function of the maximum number of random restarts allowed.

With the same approach we would be able to test the reliability of the strong Nash equilibria finding algorithm but, in general, a strong Nash equilibrium may not exists. However, by the considerations in Chapter 3 we know that random games allow only pure strategy strong Nash equilibria, we can then derive sufficient conditions for a random game to have no strong Nash equilibrium.

Proposition 5.3.1. *If a strategic form game, with random payoff values, does not have a pure strategy Nash equilibrium, then it does not have a strong Nash equilibrium.*

Moreover even if pure strategy Nash equilibria exist we are still able to determine sufficient conditions that, when satisfied, guarantee that a strong Nash equilibrium does not exist.

Proposition 5.3.2. *If in a strategic form game with random payoff values, every pure strategy Nash equilibrium is Pareto dominated by at least a different pure strategy, then a strong Nash equilibrium does not exist.*

It is clear that these conditions are sufficient but not necessary for a pure strategy to be a strong Nash equilibrium. With these two requirements we can

define the noSNE class, i.e. a class of games in which we know in advance that strong Nash equilibria does not exist. On the other hand if the problem defined by Formulation 4.4.1, *corrSNE*, is satisfiable by Theorem 5.2.3 we are sure the game has a strong Nash equilibrium. Therefore we can define define a test class hasSNE, and we know in advance that every game of this class has a strong Nash equilibrium.

We can now test the accuracy of the our algorithm in two steps. At first we check how often a solution is found when it actually exists, then we check how often it finds a strong Nash equilibrium when the game does not have any.

We define three test cases.

- Test of the accuracy of the nonlinear program used to find Nash equilibria and how it is affected by random restarts.
- Test of the accuracy of the strong Nash equilibrium algorithm in two steps:
  - with the class of games *hasSNE*, we test how the algorithm is able to find a solution given that it exists,
  - with the class of games *noSNE*, we test how often the algorithm finds a solution given that it does not exist.

However we have to consider that also the game class has an impact on the nonlinear solver, in fact the payoff values characterize the utility space. For some game classes the shape of the utility space is extremely regular, e.g. consider zero–sum games or polymatrix games, and for some other classes, as the random class, the characteristics of the utility space are not known in advance and its shape is typically irregular.

The games samples for our test sets are generated using GAMUT [21]. We generate instances of three agents games, with the same number of actions for esch agent, from the RandomGame class. The experiments are conducted on a Intel 2.20GHz processor with Linux kernel 2.6.32.

5.3.2 Test results

In Table 5.1 we present the result of the execution of SNOPT to solve the *NE* problem, Formulation 2.2.8. In the first column the number of actions per agent are reported. The second field is the number of maximum random restarts allowed. The accuracy is the percentage of games in which the nonlinear solver is able to find a Nash equilibrium. In the time field the average time to compute a Nash equilibrium is reported, notice that in general the time to compute a Nash equilibrium may be different from the SNOPT execution time, as multiple restarts may be required. In the restart column we report the average number of random restarts required to compute a Nash equilibrium. Finally in the sample size column the number of tested games is reported. Notice that when the maximum random restart parameter is set to zero the solver is executed only once without initializing a random starting strategy. Therefore in this case the time parameter matches the average time required by SNOPT to solve the *NE* problem.

From the experimental results we notice that SNOPT performances are quite bad for this kind of problem if we do not provide random start points, allowing multiple iterations. Even with a low number of different random starting point accuracy increases, in particular for the higher dimension games. Notice that the higher is the number of actions per agent the higher is the number of average restarts needed to compute a solution. With a maximum of 50 restarts from random starting points of the nonlinear solver we achieve a satisfactory accuracy, however the average computation time greatly increases.

Max restarts	Actions	Accuracy	Time	Restarts	Sample size
0 restart	2	60%	27,3 ms	-	200
	4	29%	35,2 ms	-	100
	6	16%	51,4 ms	-	160
	8	7%	116,3 ms	-	160
	10	6%	202,9 ms	-	150
10 restarts	2	75%	110,3 ms	2,8	200
	4	73%	182,33 ms	3,9	100
	6	77%	338,3 ms	4,4	160
	8	62%	643,9 ms	5,4	160
	10	52%	1692,2 ms	6,3	150
25 restarts	2	85%	216,6 ms	6,1	200
	4	92%	224,1 ms	4,8	100
	6	91%	507 ms	7,2	160
	8	89%	1043 ms	8,9	160
	10	78%	2341 ms	11,1	150
50 restarts	2	91%	312,18 ms	8,8	200
	4	95%	289,9 ms	6,5	100
	6	98%	497,4 ms	6,9	160
	8	97%	1047,9 ms	9,5	160
	10	94%	2903,9 ms	14,8	150
100 restarts	2	93%	373,1 ms	11,9	200
	4	98%	309,1 ms	6,7	100
	6	99%	619,6 ms	8,4	160
	8	99%	1240 ms	10,9	160
	10	99%	3473,2 ms	17,1	150

Table 5.1: Experimental results: properties of Nash equilibrium–finding with SNOPT

Given the previous considerations we test the algorithm setting the maximum number of restarts, for the solution of the nonlinear problems, from different random points to 50. We test now the accuracy of our algorithm on the *hasSNE* game class. For these games we used a mix of games from different classes. It turns out that when the algorithm uses the *NEPMD* model, ref. to Formulation 4.2.1, for the oracle it has a very low accuracy (always below 10%), probably due to the high number of variables subjected to the integrality constraint, therefore for the oracle implementation we focus on the *NE*, ref. to Formulation 2.2.8, and *NEKKT*, ref.

to Formulation 4.3.1, models. In Table 5.3 we report the result on tests performed on 50 games. The *NE* and *NEKKT* columns contain the fraction of games in which the algorithm was able to find a strong Nash equilibrium using for the oracle the *NE* and the *NEKKT* formulation respectively. The fields average NE depth and average NEKKT depth show the average number of levels visited by the tree–search algorithm.

Actions	NE	NEKKT	Avg. NE depth	Avg. KKT depth
2	92%	88%	2,8	1,2
4	94%	92%	3,4	1,5
6	90%	78%	3,7	1,2
8	88%	66%	5,2	1,3

Table 5.2: Experimental results: accuracy of SNE–finding algorithm, given a strong Nash equilibrium exists.

We also test the accuracy of our algorithm on the *noSNE* game class. The results on tests performed on 50 games are showed in Table 5.3. In this case the percentages reported in this table are the fraction of games in which a strong Nash equilibrium is found even if it does not exists.

Actions	NE err.	NEKKT err.	Avg. NE depth	Avg. KKT depth
2	4%	0%	2,8	1
4	6%	2%	3,5	1,1
6	6%	0%	4,2	1
8	12%	4%	4,8	1

Table 5.3: Experimental results: percentage of errors of SNE–finding algorithm, given a strong Nash equilibrium does not exists.

From this experiments we can see that if a strong Nash equilibrium exists, the algorithm has an higher accuracy when using for the oracle the nonlinear *NE* feasibility problem, rather than the *NEKKT* problem. However in this case there are more errors (false positives) when searching a strong Nash equilibrium in games that do not have one. When the algorithm uses the *NEKKT* model for the oracle the tree search algorithm typically expands only the first level, given that a candidate solution exists at that level, and seldom it reaches the third level. Whereas if we use the *NE* model for the oracle on average we obtain deeper search trees. As a consequence in general we have a lower execution time when using the *NEKKT* model, given all the other parameters. However when the dimensions of the game increase, there is in both cases a general performance degeneration.

Chapter 6

Conclusions and Future Works

In this thesis, we provided an extended study on the solutions of games, played by rational agents, that are resilient to unilateral and multilateral deviations, known as strong Nash equilibria.

The first part of this work is focused on the study of bimatrix games where a mixed–strategy Nash equilibrium exists, on their characterization and on the definition of a smoothed–complexity class for the problem of strong Nash equilibrium finding.

The computational study of strong Nash equilibrium is a challenging task. The literature provides a thorough computational characterization of Nash equilibrium, while few results are known about strong Nash equilibrium and these two solution concepts present different properties (e.g., Nash finding is a total problem, while strong Nash finding is not). The problem of finding a strong Nash equilibrium in a strategic–form game has shown to be $\mathcal{NP}$–complete when the number of agents is a constant. However a smoothed complexity class for this problem is not known.

In this work we were able to show that:

- if, in a bimatrix game a mixed–strategy strong Nash equilibrium exists, then the payoffs, restricted to the actions in the support of the equilibrium, must satisfy a restrictive property: they must lay on the same line in agents' utilities space,
- finding a strong Nash equilibrium is in smoothed–$\mathcal{P}$, admitting a deterministic support–enumeration algorithm with smoothed polynomial running time, and therefore that hard instances are isolated.;

The last result allowed us to show that, except for a null measure space of the parameters, strategic–form games admit only pure–strategy strong Nash equilibria.

In the second part of this thesis we propose a general algorithm to find strong Nash equilibria in games with three or more players. This algorithm extends the one presented in [12], which works only with two agents game, to the case with

multiple agents. The main contributions of this second part are:

- a nonlinear program for finding a Nash equilibrium that is resilient to pure strategy coalitional deviations; we showed that it is a necessary condition for a strategy to be a strong Nash equilibrium, but not sufficient,
- a nonlinear program to find Nash equilibria that satisfy Karush–Kuhn–Tucker conditions; we showed that it is necessary for a strategy to be a strong Nash equilibrium, but not sufficient,
- a nonlinear program to find Nash equilibria that are Pareto efficient for each coalition with respect to coalition correlated–strategies; we showed that it is sufficient for a strategy to be a strong Nash equilibrium, but not necessary,
- a tree search algorithm for strong Nash equilibrium finding; we leveraged our necessary conditions to obtain better oracles for use at the search tree nodes.

The problem whether there is a necessary and sufficient set of equilibrium constraints in mathematical programming fashion is left open.

Experiments showed the viability of the approach. Using the new necessary conditions in the oracle significantly reduces search tree size compared to using Nash equilibrium conditions alone. However we also showed that, even though the use of nonlinear mathematical programming tools is necessary, because the strong Nash equilibrium problem with more than two agents is nonlinear, it has drawbacks:

- in order to find a solution we may require random restarts,
- solving nonlinar problems with Sparse Nonlinear OPTimizer, whose algorithm is not complete, in general lead our algorithms to be incomplete or not sound,
- the performance of the algorithm get worse as the game size increases.

As there are many results there are also many future research directions that can be explored.

- In order to compare the performance of our algorithm with those of the one presented in [12], it is possible to use compactly representable games that do not require nonlinear programming, even with more than two agents, such as polymatrix games. Thanks to this kind of games the linear algorithm of [12], which is sound and complete, would be able to run on a three–players game, however it has to be modified in order to work on polymatrix games.
- Moreover, it would be interesting to check how the algorithm general performances change using a nonlinear solver different than SNOPT.
- The algorithm we developed has good performances, in terms of average tree growth, however the use of many random start points to solve the nonlinar feasibility problems with an adequate reliability entails a longer execution time. A possible research direction is to develop a sound and complete algorithm, able to deal with mixed strategies, and compare its time performances with the algorithm presented here.

- In this work we focused our interest on exact solutions, however it is possible to find solutions allowing a certain error, i.e. an approximate solution. A possible future research direction is to provide a study on the computational complexity, both worst case and smoothed, of approximating a strong Nash equilibrium. Moreover, for such a problem, there will be the need to study and design new algorithms.
- Furthermore, we defined the strong Nash correlated equilibrium as a special case of strong Nash equilibrium, but we did not provide an extensive analysis on its properties. A subject for future works could be a study on computational issues related to this kind of equilibrium.
- It would also be interesting to evaluate in depth the performance of both non–linear programming solvers and mathematical programming with equilibrium constraints [18] solvers.
- Finally one of the main topic still left open is to determine whether it is possible or not to derive a finite set of necessary and sufficient conditions for a strategy to be a strong Nash equilibrium.

Bibliography

[1] R. Aumann. Acceptable points in games of perfect information. *PAC J MATH*, 10:381–417, 1960.

[2] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming - Theory and Algorithms*. Cambridge University Press, 2006.

[3] M. Bläser and B. Manthey. Smoothed complexity theory. *CoRR*, abs/1202.1936, 2012.

[4] J.M. Borwein and A.S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. CMS Books in Mathematics. Springer, 2006.

[5] X. Chen, X. Deng, and S.H. Teng. Computing Nash equilibria: approximation and smoothed complexity. In *FOCS*, pages 603–612, 2006.

[6] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two–player Nash equilibria. *J ACM*, 56(3):14:1–14:57, 2009.

[7] V. Conitzer and T. Sandholm. New complexity results about Nash equilibria. *GAME ECON BEHAV*, 63(2):621–641, 2008.

[8] C. Daskalakis, P. Goldberg, and C. Papadimitriou. The complexity of computing a Nash equilibrium. In *STOC*, pages 71–78, 2006.

[9] A. Epstein, M. Feldman, and Y. Mansour. Strong equilibrium in cost sharing connection games. In *ACM EC*, pages 84–92, 2007.

[10] R. Fourer, D.M. Gay, and B.W. Kernighan. A modeling language for mathematical programming. *Management Science*, 36(5):519–554, 1990.

[11] N. Gatti, G. Patrini, M. Rocco, and T. Sandholm. Combining local search techniques and path following for bimatrix games. In *UAI*, pages 286–295, 2012.

[12] N. Gatti, M. Rocco, and T. Sandholm. On the verification and computation of strong Nash equilibrium. In *AAMAS*, 2013.

[13] L. Gourvès and J. Monnot. On strong equilibria in the max cut game. In *WINE*, pages 608–615, 2009.

[14] A. Hayrapetyan, E. Tardos, and T. Wexler. The effect of collusion in congestion games. In *STOC*, pages 89–98, 2006.

[15] M. Hoefer and A. Skopalik. On the complexity of Pareto–optimal Nash and strong equilibria. In *SAGT*, pages 312–322, 2010.

[16] R. Holzman and N. Law-Yone. Strong equilibrium in congestion games. *GAME ECON BEHAV*, 21:85–101, 1997.

[17] C.E. Lemke and J.J.T. Howson. Equilibrium points of bimatrix games. *SIAM J APPL MATH*, 12(2):413–423, 1964.

[18] Z.-Q. Luo, J.-S. Pang, and D. Ralph. *Mathematical Programs with Equilibrium Constraints*. Cambridge University Press, 1996.

[19] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[20] R. Nessah and G. Tian. On the existence of strong Nash equilibria. *IESEG School of Management, Working Paper*, 2012.

[21] E. Nudelman, J. Wortman, K. Leyton-Brown, and Y. Shoham. Run the GAMUT: A comprehensive approach to evaluating game–theoretic algorithms. In *AAMAS*, pages 880–887, 2004.

[22] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. *GAME ECON BEHAV*, 63:642–662, 2009.

[23] O. Rozenfeld and M. Tennenholtz. Strong and correlated strong equilibria in monotone congestion games. In *WINE*, pages 74–86, 2006.

[24] T. Sandholm, A. Gilpin, and V. Conitzer. Mixed–integer programming methods for finding Nash equilibria. In *AAAI*, pages 495–501, 2005.

[25] R. Savani and B. von Stengel. Hard–to–solve bimatrix games. *ECONOMETRICA*, 74(2):397–429, 2006.

[26] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems - Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[27] Stanford Business Software Inc. http://www.sbsi-sol-optimize.com/.

[28] D. R. Martin Thompson, S. Leung, and K. Leyton-Brown. Computing Nash equilibria of action–graph games via support enumeration. In *WINE*, pages 338–350, 2011.

Appendix A

AMPL Models for Three–Agents Games

Nash equilibrium resilient to pure multilateral deviations

```
###  SETS  ###
set A1;
set A2;
set A3;

###  VARIABLES  ###
#Strategy variables
var x1{A1};
var x2{A2};
var x3{A3};

#Nash Equilibrium NLCP variables
var v1;
var v2;
var v3;

#PMD Resilience binary grand coalition variables
rgc1{A1,A2,A3};
rgc2{A1,A2,A3};
rgc3{A1,A2,A3};

#PMD Resilience binary agent 1, agent 2 coalition variables
r121{A1,A2};
r122{A1,A2};
```

```
#PMD Resilience binary agent 1, agent 3 coalition variables
r131{A1,A3};
r132{A1,A3};

#PMD Resilience binary agent 2, agent 3 coalition variables
r231{A2,A3};
r232{A2,A3};

###  PARAMETERS  ###
param a{A1,A2,A3};
param b{A1,A2,A3};
param c{A1,A2,A3};
param M:=100;

###  OBJECTIVE  ###
maximize foo: 1;

###  CONSTRAINTS  ###
subject to mixedStrategy1:
   sum{i in A1} x1[i] = 1;
subject to mixedStrategy2:
   sum{j in A2} x2[j] = 1;
subject to mixedStrategy3:
   sum{k in A3} x3[k] = 1;
subject to mixedStrategy4{i in A1}:
   x1[i] >= 0;
subject to mixedStrategy5{j in A2}:
   x2[j] >= 0;
subject to mixedStrategy6{k in A2}:
   x3[k] >= 0;

subject to NashEquilibrium1 {i in A1}:
   v1 - sum{j in A2, k in A3} a[i,j,k]*x2[j]*x3[k] >= 0;
subject to NashEquilibrium2 {j in A2}:
   v2 - sum{i in A1, k in A3} b[i,j,k]*x1[i]*x3[k] >= 0;
subject to NashEquilibrium3 {k in A3}:
   v3 - sum{i in A1, j in A2} c[i,j,k]*x1[i]*x2[j] >= 0;
subject to NashEquilibrium4 {i in A1}:
   x1[i]*(v1 - sum{j in A2, k in A3} a[i,j,k]*x2[j]*x3[k])=0;
subject to NashEquilibrium5 {j in A2}:
   x2[j]*(v2 - sum{i in A1, k in A3} b[i,j,k]*x1[i]*x3[k])=0;
subject to NashEquilibrium6 {k in A3}:
   x3[k]*(v3 - sum{i in A1, j in A2} c[i,j,k]*x1[i]*x2[j])=0;

#### PMD Grand coalition ####
```

```
subject to BinaryGrandCoalition1{i in A1, j in A2, k in A3}:
   rgc1[i,j,k] >= 0;
subject to BinaryGrandCoalition2{i in A1, j in A2, k in A3}:
   rgc2[i,j,k] >= 0;
subject to BinaryGrandCoalition3{i in A1, j in A2, k in A3}:
   rgc3[i,j,k] >= 0;
subject to BinaryGrandCoalition4{i in A1, j in A2, k in A3}:
   rgc1[i,j,k]*(1-rgc1[i,j,k]) = 0;
subject to BinaryGrandCoalition5{i in A1, j in A2, k in A3}:
   rgc2[i,j,k]*(1-rgc2[i,j,k]) = 0;
subject to BinaryGrandCoalition6{i in A1, j in A2, k in A3}:
   rgc3[i,j,k]*(1-rgc3[i,j,k]) = 0;
subject to PMDGrandCoalition1{i in A1, j in A2, k in A3}:
   v1 - a[i,j,k] >=
   -M*(3-rgc1[i,j,k]-rgc2[i,j,k]-rgc3[i,j,k]);
subject to PMDGrandCoalition2{i in A1, j in A2, k in A3}:
   v2 - b[i,j,k] >=
   -M*(3-rgc1[i,j,k]-rgc2[i,j,k]-rgc3[i,j,k]);
subject to PMDGrandCoalition3{i in A1, j in A2, k in A3}:
   v3 - c[i,j,k] >=
   -M*(3-rgc1[i,j,k]-rgc2[i,j,k]-rgc3[i,j,k]);
subject to PMDGrandCoalition4{i in A1, j in A2, k in A3}:
   v1 - a[i,j,k] >= -M*rgc1[i,j,k];
subject to PMDGrandCoalition5{i in A1, j in A2, k in A3}:
   v2 - b[i,j,k] >= -M*rgc2[i,j,k];
subject to PMDGrandCoalition6{i in A1, j in A2, k in A3}:
   v3 - c[i,j,k] >= -M*rgc3[i,j,k];

#### PMD agent 1, agent 2 coalition ####
subject to Binary12Coalition1{i in A1, j in A2}:
   r121[i,j] >= 0;
subject to Binary12Coalition2{i in A1, j in A2}:
   r122[i,j] >= 0;
subject to Binary12Coalition3{i in A1, j in A2}:
   r121[i,j]*(1-r121[i,j]) = 0;
subject to Binary12Coalition4{i in A1, j in A2}:
   r122[i,j]*(1-r122[i,j]) = 0;
subject to PMD12Coalition1{i in A1, j in A2}:
   v1 - sum{k in A3} a[i,j,k]*x3[k] >=
   -M*(2-r121[i,j]-r122[i,j]);
subject to PMD12Coalition2{i in A1, j in A2}:
   v2 - sum{k in A3} b[i,j,k]*x3[k] >=
   -M*(2-r121[i,j]-r122[i,j]);
subject to PMD12Coalition3{i in A1, j in A2}:
   v1 - sum{k in A3} a[i,j,k]*x3[k] >= -M*r121[i,j];
```

```
subject to PMD12Coalition4{i in A1, j in A2}:
   v2 - sum{k in A3} b[i,j,k]*x3[k] >= -M*r122[i,j];

#### PMD agent 1, agent 3 coalition ####
subject to Binary13Coalition1{i in A1, k in A3}:
   r131[i,k] >= 0;
subject to Binary13Coalition2{i in A1, k in A3}:
   r132[i,k] >= 0;
subject to Binary13Coalition3{i in A1, k in A3}:
   r131[i,k]*(1-r131[i,k]) = 0;
subject to Binary13Coalition4{i in A1, k in A3}:
   r132[i,k]*(1-r132[i,k]) = 0;
subject to PMD13Coalition1{i in A1, k in A3}:
   v1 - sum{j in A2} a[i,j,k]*x2[j] >=
   -M*(2-r131[i,k]-r132[i,k]);
subject to PMD13Coalition2{i in A1, k in A3}:
   v3 - sum{j in A2} c[i,j,k]*x2[j] >=
   -M*(2-r131[i,k]-r132[i,k]);
subject to PMD13Coalition3{i in A1, k in A3}:
   v1 - sum{j in A2} a[i,j,k]*x2[j] >= -M*r131[i,k];
subject to PMD13Coalition4{i in A1, k in A3}:
   v3 - sum{j in A2} c[i,j,k]*x2[j] >= -M*r132[i,k];

#### PMD agent 2, agent 3 coalition ####
subject to Binary23Coalition1{j in A2, k in A3}:
   r231[j,k] >= 0;
subject to Binary23Coalition2{j in A2, k in A3}:
   r232[j,k] >= 0;
subject to Binary23Coalition3{j in A2, k in A3}:
   r231[j,k]*(1-r231[j,k]) = 0;
subject to Binary23Coalition4{j in A2, k in A3}:
   r232[j,k]*(1-r232[j,k]) = 0;
subject to PMD23Coalition1{j in A2, k in A3}:
   v2 - sum{i in A1} b[i,j,k]*x1[i] >=
   -M*(2-r231[j,k]-r232[j,k]);
subject to PMD23Coalition2{j in A2, k in A3}:
   v3 - sum{i in A1} c[i,j,k]*x1[i] >=
   -M*(2-r231[j,k]-r132[j,k]);
subject to PMD23Coalition3{j in A2, k in A3}:
   v2 - sum{i in A1} b[i,j,k]*x1[i] >= -M*r231[j,k];
subject to PMD23Coalition4{j in A2, k in A3}:
   v3 - sum{i in A1} c[i,j,k]*x1[i] >= -M*r232[j,k];
```

Correlated strategy strong Nash equilibrium

```
###  SETS  ###
set A1;
set A2;
set A3;

###  VARIABLES  ###
#Strategy variables
var x1{A1};
var x2{A2};
var x3{A3};

#Nash Equilibrium NLCP variables
var v1;
var v2;
var v3;

#Correlated strategy variables Grand Coalition
var xcgc{A1,A2,A3};
var ugc;
var lambdagc1>=0;
var lambdagc2>=0;
var lambdagc3>=0;

#Correlated strategy variables agent 1, agent 2 Coalition
var xc12{A1,A2};
var u12;
var lambda121>=0;
var lambda122>=0;

#Correlated strategy variables agent 1, agent 3 Coalition
var xc13{A1,A3};
var u13;
var lambda131>=0;
var lambda132>=0;

#Correlated strategy variables agent 2, agent 3 Coalition
var xc23{A2,A3};
var u23;
var lambda231>=0;
var lambda232>=0;

###  PARAMETERS  ###
param a{A1,A2,A3};
```

```
param b{A1,A2,A3};
param c{A1,A2,A3};

###  OBJECTIVE  ###
maximize foo: 1;

###  CONSTRAINTS  ###
subject to mixedStrategy1:
   sum{i in A1} x1[i] = 1;
subject to mixedStrategy2:
   sum{j in A2} x2[j] = 1;
subject to mixedStrategy3:
   sum{k in A3} x3[k] = 1;
subject to mixedStrategy4{i in A1}:
   x1[i] >= 0;
subject to mixedStrategy5{j in A2}:
   x2[j] >= 0;
subject to mixedStrategy6{k in A2}:
   x3[k] >= 0;

subject to NashEquilibrium1 {i in A1}:
   v1 - sum{j in A2, k in A3} a[i,j,k]*x2[j]*x3[k] >= 0;
subject to NashEquilibrium2 {j in A2}:
   v2 - sum{i in A1, k in A3} b[i,j,k]*x1[i]*x3[k] >= 0;
subject to NashEquilibrium3 {k in A3}:
   v3 - sum{i in A1, j in A2} c[i,j,k]*x1[i]*x2[j] >= 0;
subject to NashEquilibrium4 {i in A1}:
   x1[i]*(v1 - sum{j in A2, k in A3} a[i,j,k]*x2[j]*x3[k])=0;
subject to NashEquilibrium5 {j in A2}:
   x2[j]*(v2 - sum{i in A1, k in A3} b[i,j,k]*x1[i]*x3[k])=0;
subject to NashEquilibrium6 {k in A3}:
   x3[k]*(v3 - sum{i in A1, j in A2} c[i,j,k]*x1[i]*x2[j])=0;

### Grand Coalition ###
subject to MixedToGrandCoalitionCorr{i in A1,j in A2,k in A3}:
   xcgc[i,j,k] = x1[i]*x2[j]*x3[k];
subject to corrGrandCoalition1{i in A1, j in A2, k in A3}:
   xcgc[i,j,k]>=0;
subject to corrGrandCoalition2:
   sum{i in A1, j in A2, k in A3} xcgc[i,j,k] = 1;
subject to corrGrandCoalition3:
   lambdagc1+lambdagc2+lambdagc3=1;
subject to corrGrandCoalition4{i in A1, j in A2, k in A3}:
   ugc>=lambdagc1*a[i,j,k]+lambdagc2*b[i,j,k]+
      lambdagc3*c[i,j,k];
```

```
subject to corrGrandCoalition5{i in A1, j in A2, k in A3}:
   xcgc[i,j,k]*(ugc-lambdagc1*a[i,j,k]+lambdagc2*b[i,j,k]
      +lambdagc3*c[i,j,k])=0;

### Agent 1, agent 2 Coalition ###
subject to MixedTo12CoalitionCorr{i in A1, j in A2}:
   xc12[i,j] = x1[i]*x2[j];
subject to corr12Coalition1{i in A1, j in A2}:
   xc12[i,j]>=0;
subject to corr12Coalition2:
   sum{i in A1, j in A2} xc12[i,j] = 1;
subject to corr12Coalition3:
   lambda121+lambda122=1;
subject to corr12Coalition4{i in A1, j in A2}:
   u12>=lambda121*(sum {k in A3} (a[i,j,k]*x3[k]))+
      lambda122*(sum {k in A3} (b[i,j,k]*x3[k]));
subject to corr12Coalition5{i in A1, j in A2}:
   xc12[i,j]*(u12-lambda121*(sum {k in A3} (a[i,j,k]*x3[k]))+
      lambda122*(sum {k in A3} (b[i,j,k]*x3[k])))=0;

### Agent 1, agent 3 Coalition ###
subject to MixedTo13CoalitionCorr{i in A1, k in A3}:
   xc13[i,k] = x1[i]*x3[k];
subject to corr13Coalition1{i in A1, k in A3}:
   xc13[i,k]>=0;
subject to corr13Coalition2:
   sum{i in A1, k in A3} xc13[i,k] = 1;
subject to corr13Coalition3:
   lambda131+lambda132=1;
subject to corr13Coalition4{i in A1, k in A3}:
   u13>=lambda131*(sum {j in A2} (a[i,j,k]*x2[j]))+
      lambda132*(sum {j in A2} (c[i,j,k]*x2[j]));
subject to corr13Coalition5{i in A1, k in A3}:
   xc13[i,k]*(u13-lambda131*(sum {j in A2} (a[i,j,k]*x2[j]))+
      lambda132*(sum {j in A2} (c[i,j,k]*x2[j])))=0;

### Agent 2, agent 3 Coalition ###
subject to MixedTo23CoalitionCorr{j in A2, k in A3}:
   xc23[j,k] = x2[j]*x3[k];
subject to corr23Coalition1{j in A2, k in A3}:
   xc23[j,k]>=0;
subject to corr23Coalition2:
   sum{j in A2, k in A3} xc23[j,k] = 1;
subject to corr23Coalition3:
   lambda231+lambda232=1;
```

```
subject to corr23Coalition4{j in A2, k in A3}:
   u23>=lambda231*(sum {i in A1} (b[i,j,k]*x1[i]))+
      lambda232*(sum {i in A1} (c[i,j,k]*x1[i]));
subject to corr23Coalition5{j in A2, k in A3}:
   xc23[j,k]*(u23-lambda231*(sum {i in A1} (b[i,j,k]*x1[i]))+
      lambda232*(sum {i in A1} (c[i,j,k]*x1[i])))=0;
```

Nash equilibrium and Karush–Kuhn–Tucker conditions

```
###  SETS  ###
set A1;
set A2;
set A3;

###  VARIABLES  ###
#Strategy variables
var x1{A1};
var x2{A2};
var x3{A3};

#Nash Equilibrium NLCP variables
var v1;
var v2;
var v3;

#Karush-Kuhn-Tucker grand coalition variables
var l11>=0;
var l12>=0;
var l13>=0;
var nu11;
var nu12;
var nu13;
var mu11{A1};
var mu12{A2};
var mu13{A3};

#Karush-Kuhn-Tucker agent 1, agent 2 coalition variables
var l21>=0;
var l22>=0;
var nu21;
var nu22;
var mu21{A1};
var mu22{A2};
```

```
#Karush-Kuhn-Tucker agent 1, agent 3 coalition variables
var l31>=0;
var l33>=0;
var nu31;
var nu33;
var mu31{A1};
var mu33{A3};

#Karush-Kuhn-Tucker agent 2, agent 3 coalition variables
var l42>=0;
var l43>=0;
var nu42;
var nu43;
var mu42{A2};
var mu43{A3};

###  PARAMETERS  ###
param a{A1,A2,A3};
param b{A1,A2,A3};
param c{A1,A2,A3};

###  OBJECTIVE  ###
maximize foo: 1;

###  CONSTRAINTS  ###
subject to mixedStrategy1:
   sum{i in A1} x1[i] = 1;
subject to mixedStrategy2:
   sum{j in A2} x2[j] = 1;
subject to mixedStrategy3:
   sum{k in A3} x3[k] = 1;
subject to mixedStrategy4{i in A1}:
   x1[i] >= 0;
subject to mixedStrategy5{j in A2}:
   x2[j] >= 0;
subject to mixedStrategy6{k in A2}:
   x3[k] >= 0;

subject to NashEquilibrium1 {i in A1}:
   v1 - sum{j in A2, k in A3} a[i,j,k]*x2[j]*x3[k] >= 0;
subject to NashEquilibrium2 {j in A2}:
   v2 - sum{i in A1, k in A3} b[i,j,k]*x1[i]*x3[k] >= 0;
subject to NashEquilibrium3 {k in A3}:
   v3 - sum{i in A1, j in A2} c[i,j,k]*x1[i]*x2[j] >= 0;
```

```
subject to NashEquilibrium4 {i in A1}:
   x1[i]*(v1 - sum{j in A2, k in A3} a[i,j,k]*x2[j]*x3[k])=0;
subject to NashEquilibrium5 {j in A2}:
   x2[j]*(v2 - sum{i in A1, k in A3} b[i,j,k]*x1[i]*x3[k])=0;
subject to NashEquilibrium6 {k in A3}:
   x3[k]*(v3 - sum{i in A1, j in A2} c[i,j,k]*x1[i]*x2[j])=0;

#### KKT Grand coalition ####
subject to KKTGrandCoalition1 {i in A1}:
   mu11[i]*(x1[i])=0;
subject to KKTGrandCoalition2 {j in A2}:
   mu12[j]*(x2[j])=0;
subject to KKTGrandCoalition3 {k in A3}:
   mu13[k]*(x3[k])=0;
subject to KKTGrandCoalition4:
   l11+l12+l13=1;
subject to KKTGrandCoalition5 {i in A1}:
   l11*(sum{j in A2, k in A3} a[i,j,k]*x2[j]*x3[k]) +
   l12*(sum{j in A2, k in A3} b[i,j,k]*x2[j]*x3[k]) +
   l13*(sum{j in A2, k in A3} c[i,j,k]*x2[j]*x3[k]) +
   mu11[i] = nu11;
subject to KKTGrandCoalition6 {j in A2}:
   l11*(sum{i in A1, k in A3} a[i,j,k]*x1[i]*x3[k]) +
   l12*(sum{i in A1, k in A3} b[i,j,k]*x1[i]*x3[k]) +
   l13*(sum{i in A1, k in A3} c[i,j,k]*x1[i]*x3[k]) +
   mu12[j] = nu12;
subject to KKTGrandCoalition7 {k in A3}:
   l11*(sum{i in A1, j in A2} a[i,j,k]*x1[i]*x2[j]) +
   l12*(sum{i in A1, j in A2} b[i,j,k]*x1[i]*x2[j]) +
   l13*(sum{i in A1, j in A2} c[i,j,k]*x1[i]*x2[j]) +
   mu13[k] = nu13;
subject to KKTGrandCoalition8 {i in A1}:
   mu11[i]>=0;
subject to KKTGrandCoalition9 {j in A2}:
   mu12[j]>=0;
subject to KKTGrandCoalition10 {k in A3}:
   mu13[k]>=0;

#### KKT agent 1, agent 2 coalition ####
subject to KKT12Coalition1 {i in A1}:
   mu21[i]*(x1[i])=0;
subject to KKT12Coalition2 {j in A2}:
   mu22[j]*(x2[j])=0;
subject to KKT12Coalition3:
   l21+l22=1;
```

```
subject to KKT12Coalition4 {i in A1}:
   l21*(sum{j in A2, k in A3} a[i,j,k]*x2[j]*x3[k]) +
   l22*(sum{j in A2, k in A3} b[i,j,k]*x2[j]*x3[k]) +
   mu21[i] = nu21;
subject to KKT12Coalition5 {j in A2}:
   l21*(sum{i in A1, k in A3} a[i,j,k]*x1[i]*x3[k]) +
   l22*(sum{i in A1, k in A3} b[i,j,k]*x1[i]*x3[k]) +
   mu22[j] = nu22;
subject to KKT12Coalition6 {i in A1}:
   mu21[i]>=0;
subject to KKT12Coalition7 {j in A2}:
   mu22[j]>=0;

#### KKT agent 1, agent 3 coalition ####
subject to KKT13Coalition1 {i in A1}:
   mu31[i]*(x1[i])=0;
subject to KKT13Coalition2 {k in A3}:
   mu33[k]*(x3[k])=0;
subject to KKT13Coalition3:
   l31+l33=1;
subject to KKT13Coalition4 {i in A1}:
   l31*(sum{j in A2, k in A3} a[i,j,k]*x2[j]*x3[k]) +
   l33*(sum{j in A2, k in A3} c[i,j,k]*x2[j]*x3[k]) +
   mu31[i] = nu31;
subject to KKT13Coalition5 {k in A3}:
   l31*(sum{i in A1, j in A2} a[i,j,k]*x1[i]*x2[j]) +
   l33*(sum{i in A1, j in A2} c[i,j,k]*x1[i]*x2[j]) +
   mu33[k] = nu33;
subject to KKT13Coalition6 {i in A1}:
   mu31[i]>=0;
subject to KKT13Coalition7 {k in A3}:
   mu33[k]>=0;

#### KKT agent 2, agent 3 coalition ####
subject to KKT23Coalition1 {j in A2}:
   mu42[j]*(x2[j])=0;
subject to KKT23Coalition2 {k in A3}:
   mu43[k]*(x3[k])=0;
subject to KKT23Coalition3:
   l42+l43=1;
subject to KKT23Coalition4 {j in A2}:
   l42*(sum{i in A1, k in A3} b[i,j,k]*x1[i]*x3[k]) +
   l43*(sum{i in A1, k in A3} c[i,j,k]*x1[i]*x3[k]) +
   mu42[j] = nu42;
subject to KKT23Coalition5 {k in A3}:
```

```
   l42*(sum{i in A1, j in A2} b[i,j,k]*x1[i]*x2[j]) +
   l43*(sum{i in A1, j in A2} c[i,j,k]*x1[i]*x2[j]) +
   mu43[k] = nu43;
subject to KKT23Coalition6 {j in A2}:
   mu42[j]>=0;
subject to KKT23Coalition7 {k in A3}:
   mu43[k]>=0;
```