Computing a Self–Confirming Equilibrium in Two–Player Extensive–Form Games

Nicola Gatti
Politecnico di Milano
Piazza Leonardo da Vinci 32
Milano, Italy
ngatti@elet.polimi.it

Fabio Panozzo
Politecnico di Milano
Piazza Leonardo da Vinci 32
Milano, Italy
panozzo@elet.polimi.it

Sofia Ceppi
Politecnico di Milano
Piazza Leonardo da Vinci 32
Milano, Italy
ceppi@elet.polimi.it

ABSTRACT

The Nash equilibrium is the most commonly adopted solution concept for non–cooperative interaction situations. However, it underlays on the assumption of *common information* that is hardly verified in many practical situations. When information is not common, the appropriate game theoretic solution concept is the *self–confirming equilibrium*. It requires that every agent plays the best response to her beliefs and that the beliefs are correct on the equilibrium path. We present, to the best of our knowledge, the first study on the computation of a self–confirming equilibrium for two–player extensive–form games. We provide algorithms, we analyze the computational complexity, and we experimentally evaluate the performance of our algorithms in terms of computational time.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence

General Terms

Algorithms, Economics

Keywords

Game Theory (cooperative and non-cooperative)

1. INTRODUCTION

Non–cooperative game theory provides elegant models and solution concepts for situations wherein rational agents can strategically interact [5]. The central solution concept is the Nash equilibrium: it defines how agents should act in settings where an agent's best strategy may depend on what the others do. One of the main drawbacks of employing the Nash equilibrium concept in many practical situations is the assumption of *common information*. That is, when information is *complete*, common information means that each agent knows the private utility values of her opponents and knows that her opponents know her private utility values and so on. When information is *uncertain*, the constraint of common information is harder: each agent must have a Bayesian prior over her opponents that must be common for all the agents. This assumption seems to be unrealistic in a large number of practical situations (e.g., negotiations).

Basically, a Nash equilibrium provides some prescriptions to the agents without explaining how agents can have formed a common prior. This prior formation process is customarily studied in the literature as a learning process in which each agent has some (generally incorrect) beliefs over the behaviors of her opponents and, by repeatedly observing the moves of the opponents, adjusts her beliefs by means of a learning algorithm. The crucial point is that, when we study the problem of finding the agents' optimal strategies incorporating the problem of prior formation, some steady states may not be Nash equilibria. Game theory provides a solution concept, called *self–confirming equilibrium* [3, 4], that is appropriate for these situations (regardless of the specific learning algorithm adopted by the agents). The basic idea behind the concept of self–confirming equilibrium (from here on SCE) is that the agents' beliefs need to be correct *only* at the information sets reached on the equilibrium path. Since the agents do not observe the behavior of their opponents off the equilibrium path, their beliefs can be incorrect at those information sets. The set of SCEs contains the set of the Nash equilibria, a Nash equilibrium being a SCE in which the beliefs are correct at every information set. While in strategic–form games, all the SCEs are also Nash equilibria (all the information sets being on the equilibrium path), this is not the case for extensive–form games. In these games, a SCE may not be a Nash equilibrium. The game theory literature provides also several refinements of SCEs to capture different situations (e.g., when a agent is drawn from a population of individuals).

In this work, we focus on two–player extensive–form games and we study the problem of computing a SCE and its refinements regardless of the specific learning algorithms adopted by the agents. SCEs have been already considered in some previous works both on learning algorithms [13, 19] and practical applications (e.g., auctions) [14], but, to the best of our knowledge, no work discusses how a SCE and its refinements can be computed. We extend the algorithms for finding a Nash equilibrium [17] to compute a SCE and we study their properties and computational complexity. Furthermore, we experimentally evaluate the computational time of our algorithms to compare their performances and to find the size of the game instances solvable within a reasonable time (10 minutes). We developed a game instance generator and we generated the game instances that are inspired to those used in [8, 18].

Cite as: Computing a Self–Confirming Equilibrium in Two–Player Extensive–Form Games, Gatti, Panozzo, Ceppi, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 981-988.

In our mind, the applications of our algorithms are two. They can be used to compute an equilibrium for a given non–cooperative problem (as it happens for Nash equilibrium), or they can be used within learning algorithms to guide the converge to an equilibrium or to evaluate their performance.

The rest of the paper is organized as follows. Section 2 introduces extensive–form games and algorithms to compute a Nash equilibrium. Section 3 presents the core of our results, discussing the algorithms for computing SCEs, while Section 4 provides experimental evaluations. Section 5 concludes the paper. Appendix A discusses how linear complementarity formulations can be solved.

2. EXTENSIVE–FORM GAMES AND SOLVING ALGORITHMS

2.1 Definitions

A finite *perfect–information* extensive–form game [17] is a tuple $(N, A, V, T, \iota, \rho, \chi, u)$, where: N is the set of n agents, A is a set of actions, V is the set of decision nodes of the game tree, T is the set of terminal nodes of the game tree, $\iota : V \to N$ is the agent function that specifies the agent that acts at a given decision node, $\rho : V \to \wp(A)$ returns the actions available to agent $\iota(w)$ at decision node w, $\chi : V \times A \to V \cup T$ assigns the next (decision or terminal) node to each pair composed of a decision node w and an action a available at w, and $u = (u_1, \dots, u_n)$ is the set of agents' utility functions where $u_i : T \to \mathbb{R}$. An extensive–form game is with *imperfect information* when some action of some agent is not perfectly observable by the agent's opponents. Formally, it is a tuple $(N, A, V, T, \iota, \rho, \chi, u, I)$ where $(N, A, V, T, \iota, \rho, \chi, u)$ is a perfect–information extensive–form game and $I = (I_1, \dots, I_n)$ with $I_i = (I_{i,1}, \dots, I_{i,k_i})$ is a partition of set $V_i = \{w \in V : \iota(w) = i\}$ with the property that $\rho(w) = \rho(w')$ whenever there exists a j for which $w, w' \in I_{i,j}$. The sets $I_{i,j}$ are called *information sets*. We focus on games with *perfect recall*, where every agent recalls all the actions undertaken by her and by the opponents (this assumption induces some constraints over I, omitted here for reason of space).

A *pure strategy* σ_i is a plan of actions specifying one action for each information set of agent i. A mixed strategy σ_i is a randomization over pure strategies (plans). An alternative representation is given by *behavioral strategies*. They are the strategies in which each agent's (potentially probabilistic) choice at each information set is made independently of the choices at other nodes. Essentially, a behavioral strategy σ_i assigns each information set $h \in I_i$ a probability distribution over the actions available at h. With perfect recall, the two representations (plans and behavioral) are equivalent.

Each agent has a *system of belief* providing her beliefs over the behavior of the opponents. We call μ_i^j the system of belief of agent i over strategy σ_j of agent j. The beliefs are *correct* if $\mu_i^j = \sigma_j$ for every i and j. We call $\mu_i = \{\mu_i^j : \text{ for all } j \neq i\}$. A pair (σ, μ), where σ is the agents' strategy profile and μ is the set of all the agents' μ_i, is called *assessment*.

Under the assumption that information is complete and common we can define the concepts of *Nash equilibrium* as an assessment (σ, μ) such that for all $i \in N$: strategy σ_i is a best response to μ_i, and beliefs μ are correct.

It is well known that in extensive–form games some Nash equilibria may be not reasonable with respect to the sequential structure of the game. The concept of *sequential equilibrium* refines the concept of Nash equilibrium removing these equilibria [9]. A sequential equilibrium is an assessment (σ, μ) such that for all $i \in N$: strategy σ_i is sequentially optimal with respect to μ_i (in the sense of backward induction), and there exists a sequence of fully mixed strategies $\tilde{\sigma}_{i,m}$ such that for all agents i $\lim_{m \to +\infty} \tilde{\sigma}_{i,m} = \sigma_i$ and the limit of the sequence of beliefs derived from the fully mixed strategies by using the Bayes rule converges to μ. The second condition is called *Kreps and Wilson consistency* and entails that the beliefs are correct. (The use of fully mixed strategies is accomplished to characterize beliefs off the equilibrium path where the Bayes rule cannot be applied.)

2.2 The sequence form

The computation of a Nash equilibrium in an extensive–form game can be easily accomplished by transforming the game in *normal form* and then by computing a Nash equilibrium. We recall that the normal-form of an extensive–form game is a matrix–based representation where the agents' actions are plans of actions in the extensive–form game. However, the normal–form is exponential in the size of the extensive–form game, making the computation of a Nash equilibrium hard. One way to avoid this problem is to work directly on the extensive–form representation by employing behavioral strategies. This can be efficiently accomplished by using an alternative representation called *sequence form* [7]. This is a sparse matrix based representation where: (i) each agent's actions are (terminal and non–terminal) sequences q of her actions in the game tree (consider Fig. 1, $q = R$ is a non–terminal sequence of agent 1, while $q = RL_1$ is terminal); (ii) given a profile of sequences $q = (q_1, \dots, q_n)$ where q_i is the sequence of agent i, if q leads to a terminal node, then the agents' payoffs are their utilities over such a node, otherwise the payoffs are null; and (iii), called $q' = q|a$ the sequence obtained by extending q with action a (e.g., $q' = RL_1$ with $q = R$ and $a = L_1$), the probability of a sequence q is equal to the sum of the probabilities of the sequences that extend it. Once a game is solved in sequence form, the behavioral strategies can be easily computed.

We report the sequence form constraints for two–player games in a mathematical programming fashion. We explicitly consider the agents' beliefs (even if they can be omitted, being correct in a Nash equilibrium) because we shall use them in the next section. We denote the probabilities with which agents make their sequences (i.e., σ) by $p_i(q)$, and we denote the agents' systems of beliefs (i.e., μ) by $\hat{p}_i(q)$, where $\hat{p}_i(q)$ is the belief of agent $-i$ over the strategy of agent i. We denote by Q_i the set of sequences of agent i, by I_q the set of the information sets of agent i reachable from sequence $q \in Q_i$ (consider Fig. 1, $I_R = \{1.2\}$), and by h_q a generic information set belonging to I_q. Strategies $p_i(\cdot)$ and beliefs $\hat{p}_i(\cdot)$ are subject to the following constraints ($\varnothing$ is the empty sequence):

$$\hat{p}_i(\varnothing) = 1 \qquad \forall i \in N \tag{1}$$

$$\hat{p}_i(q) = \sum_{a \text{ at } h_q} \hat{p}_i(q|a) \qquad \forall i \in N, q \in Q_i, h_q \in I_q \tag{2}$$

$$\hat{p}_i(q) \geq 0 \qquad \forall i \in N, q \in Q_i \tag{3}$$

$$p_i(\varnothing) = 1 \qquad \forall i \in N \tag{4}$$

$$p_i(q) = \sum_{a \text{ at } h_q} p_i(q|a) \qquad \forall i \in N, q \in Q_i, h_q \in I_q \tag{5}$$

$$p_i(q) \geq 0 \qquad \forall i \in N, q \in Q_i \tag{6}$$

We introduce two further constraints that we shall use in what follows. We denote by $v_i(q)$ the utility agent i receives from taking sequence q and by $\overline{v}_h$ the utility an agent expects to gain when it plays at information set h (i.e., the largest expected utility among those of the sequences $q|a$ where a is available at h).

$$v_i(q) = \sum_{q' \in Q_{-i}} \hat{p}_{-i}(q') U_i(q, q') + \sum_{h \in I_q} \overline{v}_h \qquad \forall i \in N, q \in Q_i \qquad (7)$$

$$v_i(q|a) \leq \overline{v}_h \qquad \begin{array}{c}\forall i \in N, q|a \in Q_i, \\ a \text{ at } h, h \in I_q\end{array} \qquad (8)$$

Constraints (7) state that the agent i's expected utility from sequence q is equal to the sum of the expected utility over the terminal outcomes (if reached) and of the expected utilities of the information sets reachable by performing q (if exist). Constraints (8) state that the utility at h is not smaller than the utility of all the sequences $q|a$ where a is available at h.

2.3 Computing an equilibrium

The computation of a Nash equilibrium is essentially a feasibility mathematical programming problem [17] that always admits at least a solution in mixed strategies. It is known to be PPAD–complete [2]. We recall that it is generally believed that PPAD≠P and that computing a Nash equilibrium requires exponential time in the worst case. For a two–player game there are three main exact solving algorithms. LH provides a linear complementarity mathematical programming formulation and an algorithm based on pivoting techniques [11]. While LH is applicable to solve an extensive–form game in normal form, it cannot be applied to solve it in sequence form. In this case, a generalization of LH, called Lemke's algorithm is commonly used [10]. SGC provides a mixed integer linear mathematical programming formulation [16]. PNS provides an algorithm based on support enumeration [15]. SGC and PNS have been never used for extensive–form games. Below we discuss how they can be extended to these games because they play a crucial role for the computation of a SCE (precisely, a subclass of SCE cannot be computed by linear complementarity programming).

We report the mathematical programming formulations of the above algorithms for the extensive–form. At first, we require that the beliefs are correct:

$$\hat{p}_i(q) = p_i(q) \qquad \forall i \in N, q \in Q_i \qquad (9)$$

The extensive–form linear complementarity mathematical programming formulation (called ELC) is:

constraints (1), (2), (3), (4), (5), (6), (7), (8), (9)

$$(\overline{v}_h - v_i(q|a)) p_i(q|a) = 0 \qquad \begin{array}{c}\forall i \in N, q|a \in Q_i, \\ a \text{ at } h, h \in I_q\end{array} \qquad (10)$$

Constraints (10) state that, if sequence $q|a$ is played with strictly positive probability, then its utility $v_i(q|a)$ must be equal to the utility $\overline{v}_h$ of the information set h at which a is played (i.e., at every $h \in I$ agent $\iota(h)$ plays her best actions).

The mixed integer linear problem (called ESCG) is based on binary variables $s_i(q) \in \{0, 1\}$ such that, when sequence q is in the support of agent i (i.e., $p_i(q) > 0$), $s_i(q) = 1$. We notice that, by the sequence form constraints, we can have that $p_i(q) > 0$ even when q is played off the equilibrium path (e.g., consider Fig. 1, when $(p_1(L) = 1, p_2(r) = 1)$, r is off the equilibrium path). The ESCG formulation is:

constraints (1), (2), (3), (4), (5), (6), (7), (8), (9)

$$\overline{v}_h \leq v_i(q|a) + M(1 - s_i(q)) \qquad \forall i \in N, q|a \in Q_i, a \text{ at } h, h \in I_q \qquad (11)$$

$$p_i(q) \leq s_i(q) \qquad \forall i \in N, q \in Q_i \qquad (12)$$

where M is an arbitrarily large constant. Constraints (11), with constraints (8), state that, if $s_i(q) = 1$, then $\overline{v}_h = v_i(q|a)$; constraints (12) state that, if $s_i(q) = 0$, then q is played with a probability of zero.

While in ESGC the check of whether an equilibrium exists with a given support and the scan of the supports are solved together in a mathematical programming fashion, in EPNS they are separated. The first problem is solved by enumeration and heuristics and the second one is formulated as a linear mathematical programming problem that is exactly the ESCG formulation in which the values of $s_i(q)$ are fixed. Essentially, every problem instance that can be formulated as an ESCG problem can be also formulated as an EPNS problem. For reasons of space, we limit our discussion to mixed–integer formulations (i.e., ESGC), it being always possible to provide a corresponding EPNS formulation.

The unique known algorithm to compute a sequential equilibrium is a variation of the Lemke's algorithm [12]. Basically, a perturbation $\epsilon \geq 0$ is introduced into the ELC formulation and a strategy satisfying the problem both for $\epsilon = 0$ and for an arbitrarily small strictly positive value $\epsilon > 0$ is found. The formulation is:

constraints (1), (2), (3), (4), (5), (6), (7), (8), (9)

$$(\overline{v}_h - v_i(q|a))(p_i(q|a) - \epsilon^{|q|}) = 0 \qquad \begin{array}{c}\forall i \in N, q|a \in Q_i, \\ a \text{ at } h, h \in I_q\end{array} \qquad (13)$$

$$p_i(q) \geq \epsilon^{|q|} \qquad \forall i \in N, q \in Q_i \qquad (14)$$

where $|q|$ is the length of q. The perturbation given by constraints (14) assures that the solution is a *quasi–perfect equilibrium* that is a concept stronger than the sequential equilibrium. The solving algorithm is described in [12] (the perturbation is used during the pivoting exclusively in the lexicographic minimum ratio test to select the variable to be dropped from the basis). Finding a sequential equilibrium is believed to be PPAD–hard.

3. SELF–CONFIRMING EQUILIBRIA AND THEIR COMPUTATION

3.1 Equilibrium concepts

The basic self–confirming equilibrium solution concept captures the situation in which agents have no *a–priori* information about opponents' strategies or payoffs and learn (in some way) from their observations over the actions played by the opponents. The aim is the study of assessments (σ, μ) that are steady states. Essentially, they generalize the concept of Nash equilibrium to the case in which information is not common. Indeed, while Nash equilibrium provides a prescription on how rational agents should play, self–confirming equilibrium provides a prescription on what are the beliefs of rational agents and on how they should play. A self–confirming equilibrium requires that agents correctly forecast the actions that the opponents will take only on the equilibrium path, an agent deriving information on her opponents' behavior only from her observations. Off the equilibrium path the agents' beliefs can be arbitrary.

Fudenberg and Levine provide some concepts of self–confirming equilibrium [3, 4]. They distinguish between *unitary* and *heterogeneous* self–confirming equilibria. The idea is that each agent can be characterized by a population of individuals and, every time the game is repeated, a specific individual plays. Potentially, different individuals can have different beliefs and different optimal strategies. In unitary SCEs (from here on USCEs), the population is composed of a single individual (therefore each agent has exactly one belief and one optimal strategy). In heterogeneous SCEs (from here on HSCEs), the population is composed of multiple individuals. We notice that this last model perfectly apply to practical economic situations, such as, e.g., bargaining and auctions, where different sellers are continuously matched with different buyers.

Fudenberg and Levine show that SEs $\subseteq$ NEs $\subseteq$ USCEs $\subseteq$ HSCEs (where SE and NE mean sequential and Nash equilibrium respectively). They provided also two refinements (applicable to both USCEs and HSCEs).

Consistent SCE captures situations in which agents are occasionally matched with "crazy" opponents, so that even if they stick to their equilibrium strategy themselves, they eventually learn the strategy at all information sets that can be reached if their opponents deviate. It requires that each agent correctly predicts the strategy at all the information sets that can be reached when the agents' opponents, but not the agents themselves, deviate from their equilibrium strategies. In each two–player game, every SCE is consistent. For the sake of presentation, we shall omit the adjective 'consistent' in what follows, our algorithms being only for two–player games.

Rationalizable SCE captures the situations in which the agents have some information about the payoffs of their opponents and use it in the sense of rationalizability. Technically speaking, it requires that the agents' strategies are sequentially rational with respect to the beliefs (as in sequential equilibria) and beliefs are correct on the equilibrium path and on the reachable information sets (i.e., the information sets that an agent can reach by perturbing its own strategy and keeping fixed the opponents' strategy).

In [4], the authors show that: an USCE may not be a Nash, SEs $\subseteq$ rationalizable USCEs, there can be rationalizable USCEs that are not NEs, and there can be NEs that are not rationalizable USCEs.

3.2 Unitary SCE

Formally, an USCE is an assessment (σ, μ) such that for every agent $i \in N$:

- strategy σ_i is optimal with respect to some μ_i,
- all the beliefs prescribed by μ_i are correct on the equilibrium path.

That is, we need to relax constraints (9), forcing $\hat{p}_i(q) = p_i(q)$ only if q is on the equilibrium path. In order to check whether or not a sequence q is on the equilibrium path we need to consider the strategies of both agents. Indeed, as discussed in Section 2.3, in the sequence form a sequence q can present $p(q) > 0$ even if it is played off the equilibrium path. Basically, a sequence q of agent i is played on the equilibrium path if and only if q is played with strictly positive probability and, called $q = q'|a$, there exists a sequence $f(q)$ of agent $-i$ played with strictly positive probability that leads to the information set where agent i plays a. Formally, $\hat{p}_i(q) = p_i(q)$ if $p_i(q) > 0$ and $p_{-i}(f(q)) > 0$ for at least a $f(q)$ (given a q there can be multiple $f(q)$), e.g., consider Fig. 1, $q = RL_1$ is on the path if $p_1(RL_1) > 0$ and $p_2(f(RL_1)) > 0$ with $f(RL_1) \in \{l\}$, and $q = l$ is on the path if $p_2(l) > 0$ and $p_1(f(l)) > 0$ with $f(l) \in \{M, R\}$.

We extend the mathematical programming formulations provided in Section 2.3 to find a USCE. At first, we consider the ELC formulation. This formulation cannot be extended to find a USCE by introducing exclusively linear complementarity constraints. This is because, checking whether or not a sequence q is on the path is intrinsically quadratic due to the presence of the operator 'and' between conditions $p_i(q) > 0$ and $p_{-i}(f(q)) > 0$. A non–linear complementarity constraint (non–solvable by Lemke's algorithm and requiring different algorithms such as Scarf's [17]) can be:

$$p_i(q)p_{-i}(f(q))(\hat{p}_i(q) - p_i(q)) = 0 \quad \forall i \in N, q \in Q_i, f(q) \in Q_{-i}$$

(We cannot exclude that an alternative linear formulation exists, anyway we have not been able to find it.) Instead, the ESCG (and, consequently, the EPNS) formulation(s) can be extended. The ESGC can be easily modified by substituting constraints (9). More precisely, the ESGC formulation finding a USCE is:

constraints (1), (2), (3), (4), (5), (6), (7), (8), (11), (12)

$$\hat{p}_i(q) \le p_i(q) + M(2 - s_{-i}(f(q)) - s_i(q)) \quad \forall i \in N, q \in Q_i \quad (15)$$
$$\hat{p}_i(q) \ge p_i(q) - M(2 - s_{-i}(f(q)) - s_i(q)) \quad \forall i \in N, q \in Q_i \quad (16)$$

Constraints (15) and (16) force beliefs to be correct when $s_{-i}(f(q)) = 1$ and $s_i(q) = 1$.

We know that $p_i(\cdot)$ and $p_{-i}(\cdot)$ may not constitute a NE (e.g., see Example 1 in Section 3.5). However, surprisingly, we have that $\hat{p}_i(\cdot)$ and $\hat{p}_{-i}(\cdot)$ constitute a NE.

THEOREM 3.1. *Given a USCE, expressed as a set of strategies $p_i(\cdot)$ and beliefs $\hat{p}_i(\cdot)$, strategies $p'_i(\cdot) = \hat{p}_i(\cdot)$ constitute a Nash equilibrium.*

Proof. By definition, on the equilibrium path, the actions played with positive probability in $p'_i(\cdot) = \hat{p}_i(\cdot)$ are best responses to $\hat{p}_{-i}(\cdot)$, $p'_i(\cdot)$ being the same of $p_i(\cdot)$. Off the equilibrium path, the actions played with positive probability in $p'_i(\cdot)$ are potentially different from those in $p_i(\cdot)$, but, providing a utility of zero, agent i cannot gain more by deviating from them. Therefore, $p'_i(\cdot) = \hat{p}_i(\cdot)$ is a best response to $\hat{p}_{-i}(\cdot)$ and then $(p'_i(\cdot), p'_{-i}(\cdot))$ constitutes a Nash equilibrium. $\square$

As a result, given a USCE, we can find a Nash equilibrium in constant time. We can state the following theorem, whose proof is a trivial application of Theorem 3.1.

COROLLARY 3.2. *For any USCE there exists a Nash equilibrium that induces the same randomization over the outcomes.*

We focus on the computational complexity of finding a USCE.

THEOREM 3.3. *The problem of computing a USCE in a two–player game (called* USCE–2*) is PPAD–complete.*

Proof. USCE–2 is in PPAD because any USCE–2 instance admits at least one solution and, given an assessment, it can be verified in polynomial time in the size of the game whether or not it is a solution. The PPAD–completeness can be proved by reduction to NASH (the problem of computing a Nash equilibrium). A trivial reduction is due to the fact that in strategic-form games every Nash equilibrium is a USCE. A less–trivial reduction is due to Theorem 3.1. □

3.3 Heterogeneous SCE

Formally, an HSCE is an assessment (σ,μ) such that for every agent $i \in N$:

- each pure strategy j in σ_i is optimal with respect to some (potentially different) μ_i (denoted by $\mu_{i,j}$),
- the beliefs prescribed by $\mu_{i,j}$ are correct on the equilibrium path identified by pure strategy j in σ_i.

According to above definition, we need to introduce different (heterogeneous) beliefs for each agent. More precisely, according [4] we define $\hat{p}_{i,q}(q')$ as the belief of agent $-i$ over the probability with which agent i plays sequence $q' \in Q_i$, where the parameter is $q \in Q_{-i}$. For each $\hat{p}_{i,q}(q')$ the sequence form constraints must hold:

$$\hat{p}_{i,q}(\varnothing) = 1 \qquad \forall i \in N, q \in Q_{-i} \tag{17}$$

$$\hat{p}_{i,q}(q') = \sum_{a \text{ at } h_{q'}} \hat{p}_i(q'|a) \qquad \forall i \in N, q' \in Q_i, q \in Q_{-i}, h_{q'} \in I_{q'} \tag{18}$$

$$\hat{p}_{i,q}(q') \geq 0 \qquad \forall i \in N, q' \in Q_i, q \in Q_{-i} \tag{19}$$

Given that the beliefs are parameterized with respect to sequence q and the expected utility of playing a sequence q' depends on the beliefs, we need to specify the parameter q in the expected utility formula. That is, we denote by $v_{i,q}(q')$ the expected utility received by agent i when she plays sequence q' and the beliefs are those parameterized with respect to sequence q (i.e., $\hat{p}_{-i,q}(\cdot)$). We can easily check whether or not a sequence q is a never best response (i.e., there is not any belief such that q is a best response). For simplicity, we safely limit to terminal sequences. A non–terminal sequence q is not a never best response if there exists at least a terminal sequence q' extending q that is not a never best response. We denote by Q^*_i the set of terminal sequences of agent i. A sequence $q \in Q^*_i$ is not a never best response if there are some $\hat{p}_{-i,q}(q'')$ such that:

$$v_{i,q}(q') = \sum_{q'' \in Q_{-i}} \hat{p}_{-i,q}(q'')U_i(q',q'') \qquad \forall i \in N, q, q' \in Q^*_i \tag{20}$$

$$v_{i,q}(q) \geq v_{i,q}(q') \qquad \forall i \in N, q, q' \in Q^*_i \tag{21}$$

According to the definition of HSCE, we need to constrain the beliefs $\hat{p}_{i,q}(\cdot)$ to which a sequence q is a best response to be correct on the equilibrium path identified by q, e.g., consider Fig. 1, beliefs $\hat{p}_{2,L}(\cdot)$ can be any, no information set of agent 2 being on the equilibrium path identified by sequence $q = L$, instead beliefs $\hat{p}_{2,M}(\cdot)$ must be correct at least at information set 2.1, this information set being on the equilibrium path identified by sequence $q = M$. We state the problem of finding a HSCE as a mixed–integer linear programming problem as follows:

constraints (4), (5), (6), (12), (17), (18), (19), (20)

$$v_{i,q}(q) \geq v_{i,q}(q') - M(1 - s_i(q)) \qquad \forall i \in N, q, q' \in Q^*_i \tag{22}$$

$$\hat{p}_{i,q}(q') \leq p_i(q') + M(1 - s_i(q')) \qquad \begin{array}{r}\forall i \in N, q' \in Q_i, q \in Q^*_{-i},\\ q' \text{ extends somehow } f(q)\end{array} \tag{23}$$

$$\hat{p}_{i,q}(q') \geq p_i(q') - M(1 - s_i(q')) \qquad \begin{array}{r}\forall i \in N, q' \in Q_i, q \in Q^*_{-i},\\ q' \text{ extends somehow } f(q)\end{array} \tag{24}$$

Constraints (22) with constraints (12) force sequences q to be played with a probability of zero if beliefs $\hat{p}_{-i,q}(\cdot)$ are such that q is not a best response. Constraints (23) and (24) force $\hat{p}_{i,q}(\cdot)$ to be correct only on the equilibrium path identified by q.

Differently from what happens for the computation of a USCE, there is a straightforward linear complementarity formulation for finding a HSCE. This is because the equilibrium path identified by a single sequence $q \in Q_i$ depends only on q and the strategy of agent $-i$, but not on the strategy of agent i. Call $\overline{v}_{i,q}$ the largest expected utility among $v_{i,q}(q')$ for all $q' \in Q_i$. The formulation is:

constraints (4), (5), (6), (17), (18), (19), (20)

$$\overline{v}_{i,q} \geq v_{i,q}(q') \qquad \forall i \in N, q \in Q^*_i, q' \in Q \tag{25}$$

$$p_i(q)(v_{i,q}(q) - \overline{v}_{i,q}) = 0 \qquad \forall i \in N, q \in Q^*_i \tag{26}$$

$$p_i(q')(\hat{p}_{i,q}(q'') - p_i(q'')) = 0 \qquad \begin{array}{r}\forall i \in N, q', q'' \in Q_i,\\ q \in Q^*_{-i}, q'' = q'|a,\\ q \text{ extends somehow } f(q')\end{array} \tag{27}$$

Constraints (25) force $\overline{v}_{i,q}$ to be the largest expected utility among $v_{i,q}(q')$ for all $q' \in Q_i$; constraints (26) force a sequence q to be played only if it is a best response to $\hat{p}_{i,q}(\cdot)$; constraints (27) force beliefs $\hat{p}_{i,q}(\cdot)$ to be correct on the equilibrium path identified by q. Rigorously speaking, constraints (27) are not expressed as linear complementarities because $\hat{p}_{i,q}(q'') - p_i(q'')$ may be negative. Anyway, calling $\hat{p}_{i,q}(q'') = \hat{p}_{i,q}(q'')^+ + \hat{p}_{i,q}(q'')^-$ and $p_i(q'') = p_i(q'')^+ + p_i(q'')^-$, we can express constraints (27) as linear complementarity constraints as $p_i(q')(\hat{p}_{i,q}(q'')^+ - p_i(q'')^+) = 0$ and $p_i(q')(p_i(q'')^- - \hat{p}_{i,q}(q'')^-) = 0$ imposing that $\hat{p}_{i,q}(q'')^+ - p_i(q'')^+ \geq 0$ and $p_i(q'')^- - \hat{p}_{i,q}(q'')^- \geq 0$. Finally, we notice that combining the USCE's constraints with the HSCE's constraints, we can capture asymmetric situations where there is a single individual for agent i and a population for agent $-i$.

We discuss the relationship between HSCEs and USCEs.

THEOREM 3.4. *An HSCE induces a randomization over outcomes that may not occur in any USCE.*

Proof. The proof is by an example. In particular, see Example 2 in Section 3.5. □

We focus on the computational complexity of HSCE (the proof is the based on the first reduction used in the proof of Theorem 3.3).

THEOREM 3.5. *The problem of computing an HSCE in a two–player game (called* HSCE–2*) is PPAD–complete.*

3.4 Rationalizable SCE

We initially consider rationalizable USCEs. Formally, a RUSCE is an assessment (σ,μ) such that for every $i \in N$:

- strategy σ_i is sequentially optimal with respect to some μ_i,

- all the beliefs prescribed by μ_i are correct on the equilibrium path and on the off the equilibrium path information sets reachable by agent i when her strategy is perturbed.

Since we must assure rationality off (a portion of) the equilibrium path, we resort to the formulation to find a sequential equilibrium. The formulation for finding a RUSCE is:

constraints (1), (2), (3), (4), (5), (6), (7), (8), (14)

$$\hat{p}_i(q) \geq \epsilon^{|q|} \quad \forall i \in N, q \in Q_i \tag{28}$$

$$(p_i(q) - \epsilon^{|q|})(\hat{p}_i(q|a) - p_i(q|a)) = 0 \quad \forall i \in N, q \in Q_i \tag{29}$$

Constraints (28) force every belief to have strictly positive probability, granting the sequential rationality with respect to the beliefs; constraints (29) force beliefs at off the equilibrium path reachable information sets to be correct. Rigorously speaking, constraints (29) are not linear complementarities because $\hat{p}_i(q|a) - p_i(q|a)$ may be negative. Anyway, these constraints can be expressed in linear complementarity fashion as we accomplished for constraints (27). We notice that combining the USCE's constraints with the RUSCE's constraints we can capture asymmetric situations where only agent i have some information over agent $-i$'s payoffs.

The authors state in [3] that the idea behind rationalizable USCEs can be extended to HSCEs, but they do not discuss how. We study this extension, showing that the sets of RUSCEs and RHSCEs are essentially the same and then the concept of rationalizable SCE does not depend on whether the equilibrium is unitary or heterogeneous. For this reason, we omit the mathematical programming formulation for finding a RHSCE. Formally, a RHSCE is an assessment (σ, μ) such that for every i:

- each pure strategy j in σ_i is sequentially optimal with respect to some (potentially different) μ_i (denoted by $\mu_{i,j}$),
- all the beliefs prescribed by $\mu_{i,j}$ are correct on the equilibrium path identified by pure strategy j in σ_i and at all the information sets reachable by agent i by perturbing her pure strategy j.

We state the following theorem that shows that the sets of RUSCEs and RHSCEs are essentially the same.

THEOREM 3.6. *Given a RHSCE (σ, μ), any assessment (σ', μ') with $\sigma' = \sigma$ and $\mu'_i = \mu_{i,j}$ for any j is a RUSCE.*

Proof. It can be easily observed that the set of information sets at which the beliefs $\mu_{i,j}$ must be correct does not depend on j. This is because, although j identifies a different equilibrium path with respect to other sequence $k \neq j$, we have that by perturbing strategy j the set containing the reachable information sets and those on the equilibrium path is the same. Then, $\mu_{i,j} = \mu_{i,k}$ on the equilibrium path and at the reachable information sets for all j, k. Beliefs $\mu_{i,j}$ can differ only at non reachable information sets, but these beliefs do not affect the computation of the agents' best response. Therefore, any assessment (σ', μ') with $\sigma' = \sigma$ and $\mu'_i = \mu_{i,j}$ for any j is a RUSCE. □

We focus on the computational complexity of finding a RSCE with two agents (the proof is trivial, the problem can be formulated as a path–following problem and a solution can be verified in polynomial time).

THEOREM 3.7. *The problem of computing a RSCE in a two–player game (called* RSCE–2*) is PPAD–complete.*

3.5 Examples

We depict in Fig. 1 an example of two–player extensive–form game with imperfect information. In what follows we report some equilibria specifying strategies $p_i(\cdot)$ and beliefs $\hat{p}_i(\cdot)$. For reasons of space, we report only the non-null probabilities.

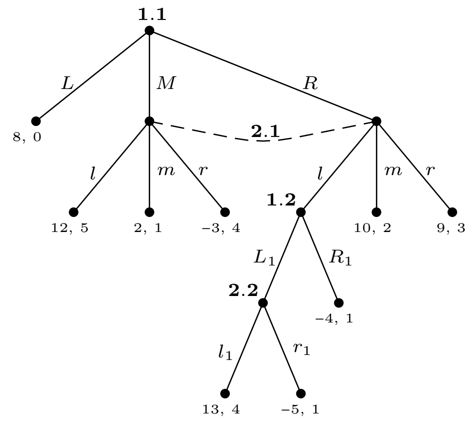

Figure 1: Example of two–player extensive–form game. ("x.y" denotes the y-th information set of agent x.)

The NEs in pure strategies are: $\sigma = (p_1(M) = 1, p_2(lr_1) = 1)$, $\sigma = (p_1(RR_1) = 1, p_2(r) = 1)$, and $\sigma = (p_1(RL_1) = 1, p_2(ll_1) = 1)$. The unique SE in pure strategies is: $\sigma = (p_1(RL_1) = 1, p_2(ll_1) = 1)$.

Example 1. A USCE that is not a NE is: $\sigma = (p_1(L) = 1, p_2(ll_1) = 1)$ with $\mu = (\hat{p}_1(L) = 1, \hat{p}_2(l) = \hat{p}_2(m) = \hat{p}_2(r) = \frac{1}{3}, \hat{p}_2(ll_1) = \frac{1}{3})$. The agents' strategies are not optimal, agent 1 gaining more by playing $q = RL_1$, while the beliefs are confirmed on the equilibrium path.

Example 2. An HSCE that is not a USCE: $\sigma = (p_1(L) = \frac{1}{2}, p_1(M) = \frac{1}{12}, p_1(RL_1) = \frac{5}{12}, p_2(ll_1) = \frac{1}{2}, p_2(r) = \frac{1}{2})$ where sequence $q = L$ is a best response to beliefs $\mu_{1,L} = (\hat{p}_{2,L}(lr_1) = \hat{p}_{2,L}(r) = \hat{p}_{2,L}(m) = \frac{1}{3})$ that are all incorrect, agent 2 playing only off the equilibrium path; sequence $q = M$ is a best response to beliefs $\mu_{1,M} = (\hat{p}_{2,M}(lr_1) = \hat{p}_{2,L}(r) = \frac{1}{2})$ that are correct on $q = l, m, r$, being on the equilibrium path, but incorrect on $q = ll_1, lr_1$, being off the equilibrium path; sequence $q = RL_1$ is a best response to beliefs $\mu_{1,RL_1} = (\hat{p}_{2,RL_1}(ll_1) = \hat{p}_{2,RL_1}(r) = \frac{1}{2})$ that are correct everywhere, all the information sets of agent 2 being on the equilibrium path; sequence $q = ll_1$ is a best response to beliefs $\mu_{2,ll_1} = (\hat{p}_{1,ll_1}(M) = \frac{1}{12}, \hat{p}_{1,ll_1}(RL_1) = \frac{5}{12})$ that is correct everywhere, all the information sets of agent 1 being on the equilibrium path; and sequence $q = r$ is a best response to beliefs $\mu_{2,r} = (\hat{p}_{1,r}(M) = \frac{1}{12}, \hat{p}_{1,r}(RR_1) = \frac{5}{12})$ that are correct on $q = L, M, R$, being on the equilibrium path, but incorrect on $q = RL_1, RR_1$, being off the equilibrium path. Notice that (M, r) does not occur in any USCE.

Example 3. A NE that is a RUSCE is $\sigma = (p_1(RR_1) = 1, p_2(r) = 1)$ with $\mu = (\hat{p}_1(RR_1) = 1, \hat{p}_2(r) = 1, \hat{p}_2(lr_1) \to 1$ in perturbation). Fixed σ_2, there is not any perturbation of agent 1 such that she can observe σ_2 at information set 2.2 and then the beliefs of agent 1 on the behavior of agent 2 at information set 2.2 can be any. Fixed σ_1, there is a perturbation of agent 2 such that she can observe σ_1 at information set 1.2 and then the beliefs of agent 2 on the behavior of agent 1 at 1.2 must be correct.

Example 4. A NE that is not a RUSCE is $\sigma = (p_1(M) = 1, p_2(lr_1) = 1)$. This is because, in perturbation agent 2 takes $q = ll_1$ instead of $q = lr_1$. It can be shown that there not exists any RUSCE when $p_1(M) = 1$. Indeed, if $p_1(M) = 1$, then, by best response, $p_2(ll_1) = 1$ ($p_2(lr_1) = 1$ is removed by perturbation). Fixed σ_2, there exists a perturbation of agent 1 such that she can observe σ_2 at information set 2.2 and then the beliefs of agent 1 on the behavior of agent 2 at 2.2 must be correct. Then, by sequential rationality, agent 1 knows that, if she takes $q = RL_1$, she gains more than taking $q = M$.

4. EXPERIMENTAL EVALUATION

Our experimental setting is constituted by a set of game instances similarly to those used in [8, 18]. We produced a number of game instances characterized by the following parameters: tree *depth* (from 1 to 8), *branching factor* (from 2 to 5), *information set density* (from 0, when all the information sets are singleton, to 1, when the game is played simultaneously by the players; we used the values: 0, 0.25, 0.5, 0.75, 1). The players alternate in the game. The payoffs are randomly generated from 0 to 1 with a uniform probability distribution. For each combination of parameters we produced 100 different game instances. In our experimental evaluations we used an UnixOS based Intel Xeon CPU 2.33 Ghz with 4 MB cache and 8 GB RAM.

We experimentally evaluate the computational time needed to find the different concepts of SCE by using both mixed–integer linear and linear complementarity mathematical programming formulations. The SCG based formulations were coded by using AMPL 8.1 [1] and solved by using CPLEX 11.0 [6]. To solve the LC based formulations, we developed an *ad-hoc* algorithm. As discussed in Section 2.3, the Lemke's algorithm is commonly used to compute a NE and a SE. Anyway, this algorithm suffers of several limitations that prevent its applicability to general linear complementarity problems. More precisely, the Lemke's algorithm presents two critical issues: it strongly suffers of numerical instability and it can fail even when the LCP admits at least one solution. An alternative method to solve a LCP, that does not suffer of the Lemke's algorithm limitations, is proposed in [18]. We implemented two variations of this algorithm to find a HSCE and a RSCE respectively. (In Appendix A, we discuss the details concerning the limitations of the Lemke's algorithm and we present our solving algorithm.) We coded our algorithms in C.

We executed the algorithms with a deadline of ten minutes (as customarily accomplished in similar evaluations [16]). Table 1 reports the average computational times (NE is computed by SCG original formulation without beliefs) spent to solve the mixed–integer linear formulations when information set density is 0.5. It can be observed that computing a HSCE is harder than computing a USCE that, in its turn, is harder than computing a NE. With different values of information set density, the computational times differ for ±20%, keeping the same profile (NE is the easiest and HSCE is the hardest). The main reason is that the size (in terms of number of variables and constraints) of the HSCE mathematical programming problem is much larger than the USCE size that is, in its turn, larger than the NE size (the variables required by NE are $O(|Q_1| + |Q_2|)$, $O(2|Q_1| + 2|Q_2|)$ by USCE and RSCE, $O(|Q_1| \cdot |Q_2|)$ by HSCE).

depth	concept	branching			
		2	3	4	5
1	NE	<0.01	<0.01	<0.01	<0.01
	USCE	<0.01	<0.01	<0.01	<0.01
	HSCE	<0.01	<0.01	<0.01	<0.01
2	NE	<0.01	<0.01	0.01	0.10
	USCE	<0.01	<0.01	0.03	0.17
	HSCE	<0.01	0.01	0.09	1.92
3	NE	<0.01	0.04	22.93	–
	USCE	<0.01	0.24	41.72	–
	HSCE	<0.01	0.41	94.37	–
4	NE	<0.01	0.83	–	–
	USCE	0.02	6.75	–	–
	HSCE	0.03	44.10	–	–
5	NE	0.02	–	–	–
	USCE	0.06	–	–	–
	HSCE	0.13	–	–	–
6	NE	0.06	–	–	–
	USCE	0.77	–	–	–
	HSCE	0.78	–	–	–
7	NE	0.08	–	–	–
	USCE	1.44	–	–	–
	HSCE	4.19	–	–	–
8	NE	1.15	–	–	–
	USCE	11.00	–	–	–
	HSCE	30.24	–	–	–

Table 1: Computational times spent to solve mixed integer linear formulations.

Table 2 reports the computational times spent to solve the linear complementarity formulations. The results confirm those previously discussed with the mixed integer linear formulations: HSCE is harder than NE. RSCE is easier than HSCE, requiring a much smaller number of variables and constraints. LC based formulations are much more efficient than SCG based formulations, but they do not allow one to find an optimal equilibrium.

depth	concept	branching			
		2	3	4	5
1	NE	<0.01	<0.01	<0.01	<0.01
	HSCE	<0.01	<0.01	<0.01	<0.01
	RSCE	<0.01	<0.01	<0.01	<0.01
2	NE	<0.01	<0.01	<0.01	<0.01
	HSCE	<0.01	<0.01	<0.01	<0.01
	RSCE	<0.01	<0.01	<0.01	<0.01
3	NE	<0.01	<0.01	<0.01	31.6
	HSCE	<0.01	<0.01	10.64	–
	RSCE	<0.01	<0.01	0.09	44.83
4	NE	<0.01	<0.01	96.27	–
	HSCE	<0.01	6.72	–	–
	RSCE	<0.01	0.05	131.54	–
5	NE	<0.01	5.23	–	–
	HSCE	<0.01	67.89	–	–
	RSCE	<0.01	6.48	–	–
6	NE	<0.01	–	–	–
	HSCE	<0.01	–	–	–
	RSCE	<0.01	–	–	–
7	NE	<0.01	–	–	–
	HSCE	<0.01	–	–	–
	RSCE	<0.01	–	–	–
8	NE	0.42	–	–	–
	HSCE	5.87	–	–	–
	RSCE	1.25	–	–	–

Table 2: Computational times spent to solve linear complementarity formulations.

5. CONCLUSIONS AND FUTURE WORKS

In a large number of practical applications, the assumption of common information is hardly verified, making the adoption of the Nash equilibrium concept not justifiable. The game theory literature provides a solution concept, i.e., self–confirming equilibrium (SCE), that appropriately captures the situation where agents are rational and form their beliefs by observing the behaviors of their opponents without having a common prior. In this paper, we provide some algorithms to compute different notions of SCE, we discuss their properties, and we evaluate their performance in terms of computational time.

In future works, we shall study the computation of SCEs when there is uncertainty both in the situations where the uncertainty is not known by the agents and apply them to economic situations. We are also interested in characterizing easy and hard games for the computation of SCEs.

6. REFERENCES

[1] AMPL Opt. LLC. http://www.ampl.com, 2010.

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

[3] E. Dekel, D. Fudenberg, and D. Levine. Payoff information and self-confirming equilibrium. *J ECON THEORY*, 89(2):165–185, 1999.

[4] D. Fudenberg and D. Levine. Self-confirming equilibrium. *ECONOMETRICA*, 61(3):523–545, 1993.

[5] D. Fudenberg and J. Tirole. *Game Theory.* The MIT Press, Cambridge, USA, 1991.

[6] ILOG Inc. http://ilog.com.sg/products/cplex, 2010.

[7] D. Koller, N. Megiddo, and B. von Stengel. Efficient computation of equilibria for extensive two-person games. *GAME ECON BEHAV*, 14(2):220–246, 1996.

[8] D. Koller and A. Pfeffer. Representations and solutions for game-theoretic problems. *ARTIF INTELL*, 94(1-2):167–215, 1997.

[9] D. Kreps and R. Wilson. Sequential equilibria. *ECONOMETRICA*, 50(4):863–894, 1982.

[10] C. Lemke. Some pivot schemes for the linear complementarity problem. *MATH PROGRAM STUD*, 7:15–35, 1978.

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

[12] P. Miltersen and T. Sorensen. Computing sequential equilibria for two-player games. In *SODA*, pages 107–116, 2006.

[13] D. Monderer and M. Tennenholtz. Learning equilibrium as a generalization of learning to optimize. *ARTIF INTELL*, 171(7):448–452, 2007.

[14] A. Osepayshvili, M. Wellman, D. Reeves, and J. Mackie-mason. Self-confirming price prediction for bidding in simultaneous ascending auctions. In *UAI*, pages 441–449, 2005.

[15] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. In *AAAI*, pages 664–669, 2004.

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

[17] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations.* Cambridge University Press, Cambridge, USA, 2008.

[18] B. von Stengel, A. van den Elzen, and D. Talman. Computing normal form perfect equilibria for extensive two-person games. *ECONOMETRICA*, 70(2):693–715, 2002.

[19] M. Wellman and J. Hu. Conjectural equilibrium in multiagent learning. *MACH LEARN*, 33(2-3):179–200, 1998.

APPENDIX

A. MLCP FORMULATION

Given variables $z, w \in \mathbb{R}^n$, and coefficients square matrix $M(n,n)$ and vector $b \in \mathbb{R}^n$, a standard LCP is expressed as:

$$z, w \geq 0 \tag{30}$$

$$w = Mz - b \tag{31}$$

$$z^T \cdot w = 0 \tag{32}$$

The Lemke's algorithm is granted to terminate when matrix M and vector b satisfies two conditions: M is positive semi–definite and b is such that, if $z \geq 0$ and $Mz \geq 0$ and $z^T M z = 0$, then $z^T b \geq 0$. While a straightforward formulation satisfying these two conditions can be found to compute a NE and a SE, we were not able to find a formulation for HSCE and RSCE.

The algorithm described in [18] is granted to terminate when applied to game instances and it does not require additional conditions. It is based on mixed linear complementarity problems (MLCP). A MLCP is a generalization of a LCP, being the combination of a LCP and linear equation system. Its standard form is expressed as:

$$z_1, w \geq 0 \tag{33}$$

$$w = M_{1,1} z_1 + M_{1,2} z_2 - b_1 \tag{34}$$

$$0 = M_{2,1} z_1 + M_{2,2} z_2 - b_2 \tag{35}$$

$$z_1^T \cdot w = 0 \tag{36}$$

According to [18], the resolution of the MLCP is accomplished into two phases. In the first phase, a basis satisfying constraints (35) and (36) that is a well–defined strategy is found. In the second phase, the complementarity pivoting is applied as prescribed by the Lemke's algorithm. During the pivoting the algorithm is proved to move on well–defined strategies and not to cycle and therefore it always terminates producing a solution. We provide the MLCP formulation for finding a RSCE (the formulation for HSCE is analogous):

$$z_1 = [p_1^+, p_1^-, p_2^+, p_2^-, t_1^+, t_1^-, t_2^+, t_2^-] \qquad z_2 = [v_1, v_2, \hat{p}_1^+, \hat{p}_1^-, \hat{p}_2^+, \hat{p}_2^-]$$

$$b_1 = [0,0,0,0,0,0,0,0] \qquad b_2 = [s_1, s_2, s_1, s_2, 0,0,0,0]$$

$$M_{1,1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -F_1 & 0 & 0 & 0 \\ 0 & F_1 & 0 & 0 \\ 0 & 0 & -F_2 & 0 \\ 0 & 0 & 0 & F_2 \end{bmatrix}, M_{1,2} = \begin{bmatrix} S_1^T & 0 & 0 & 0 & -U1 & -U1 \\ S_1^T & 0 & 0 & 0 & -U1 & -U1 \\ 0 & S_2^T & -U_2^T & -U_2^T & 0 & 0 \\ 0 & S_2^T & -U_2^T & -U_2^T & 0 & 0 \\ 0 & 0 & F_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -F_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & F_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -F_2 \end{bmatrix}$$

$$M_{2,1} = \begin{bmatrix} S_1 & S_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & S_2 & S_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_1 & T_1 & 0 & 0 & -I & 0 & 0 & 0 \\ T_1 & T_1 & 0 & 0 & 0 & -I & 0 & 0 \\ 0 & 0 & T_2 & T_2 & 0 & 0 & -I & 0 \\ 0 & 0 & T_2 & T_2 & 0 & 0 & 0 & -I \end{bmatrix}, M_{2,2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ S_1 & S_1 & 0 & 0 \\ 0 & 0 & S_2 & S_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where variables t_i are auxiliaries, S_i and s_i code the sequence form constraints (1), (2), (4), (5) as described in [7], F_i and T_i code constraints (27). For reasons of space, in matrices M we report only non-zero columns. Additional constraints are $p_i^+, p_i^-, \hat{p}_i^+, \hat{p}_i^+ \geq l_i(\epsilon)$ where $l_i(\epsilon)$ is the perturbation defined as prescribed in Section 3.4. Perturbed variables are substituted as follows $\pi_i^{\pm} = p_i^{\pm} - l_i(\epsilon)$ and $\hat{\pi}_i^{\pm} = \hat{p}_i^{\pm} - l_i(\epsilon)$ and the problem is solved in π.

The initial solution is calculated similarly as accomplished in [18]. Instead, we need to modify the pivoting for what concerns the dropping variable. More precisely, the dropping variables must assure that $\hat{\pi}_i^{\pm}$ keeps to be non–negative.