Rationality authority for provable rational behavior

Rationality Authority for Provable Rational Behavior

Shlomi Dolev[1(✉)], Panagiota N. Panagopoulou[2], Mikaël Rabie[1], Elad M. Schiller[3], and Paul G. Spirakis[2,4]

[1] Ben-Gurion University of the Negev, Beer Sheva, Israel
{dolev,rabie}@cs.bgu.ac.il
[2] Computer Technology Institute, and Press "Diophantus", Rio, Greece
{panagopp,spirakis}@cti.gr
[3] Chalmers University of Technology, Gothenburg, Sweden
elad@chalmers.se
[4] University of Liverpool, Liverpool, UK
P.Spirakis@liverpool.ac.uk

Abstract. Players in a game are assumed to be totally rational and absolutely smart. However, in reality all players may act in non-rational ways and may fail to understand and find their best actions. In particular, participants in social interactions, such as lotteries and auctions, cannot be expected to always find by themselves the "best-reply" to any situation. Indeed, *agents* may consult with others about the possible outcome of their actions. It is then up to the counselee to assure the rationality of the consultant's advice. We present a distributed computer system infrastructure, named *rationality authority*, that allows safe consultation among (possibly biased) parties. The parties' advices are adapted only after verifying their feasibility and optimality by standard formal proof checkers. The *rationality authority* design considers computational constraints, as well as privacy and security issues, such as verification methods that do not reveal private preferences. Some of the techniques resembles zero-knowledge proofs. A non-cooperative game is presented by the *game inventor* along with its (possibly intractable) equilibrium. The *game inventor* advises playing by this equilibrium and offers a checkable proof for the equilibrium feasibility and optimality. Standard *verification procedures*, provided by trusted (according to their reputation) *verification procedures*, are used to verify the proof. Thus, the proposed *rationality authority* infrastructure facilitates the applications of game theory in several important real-life scenarios by the use of computing systems.

1 Introduction

Game theory is based on the assumption that (at least, some) players play rationally. This assumption is questionable in the face of the sophistication for obtaining the best strategy in (even simple) games. Thus, the application of game theory in real life is limited by the degree in which the players (who are rarely

C. Zaroliagis et al. (Eds.): Spirakis Festschrift, LNCS 9295, pp. 33–48, 2015.
DOI: 10.1007/978-3-319-24024-4_5

mathematicians, economic experts, or computer scientists) can understand the meaning of the game rules and the way to act. One famous example is auctions where every variant of an auction introduces the need for a new proof that, say, reconfirms that the second price auction is the best to use [5,22]. We have in mind a framework that will let the ordinary and inexperienced Joe and Jane safely figure their best-reply.

Distributed computer systems can implement the *rationality authority* framework that in turn, can enable (and ensure by audit schemes) rational behavior, without sacrificing the players' privacy, e.g., keeping the individual preferences (utilities) private. The framework, as depicted in Fig. 1, includes:

- The *game inventor*, which may possibly gain revenues from the game. We consider *game inventors* that create games for which they could predict the best-reply and prove their feasibility and optimality to the players/*agents*.
- The *agents* are the game participants, for which they receive verifiable advices on the feasible and optimal actions to take.
- The *verifiers* are trustable service providers that profit from selling general purpose *verification procedures*, $v()$, (using formal methods and languages), and therefore would like to have a good long-lasting reputation on being honest in checking (a variety of) proofs. We note the possibility of having several *verifiers*, such that their majority is trusted. The reputation of the *verifiers* can be updated according to the (majority of their) results.

The *verification procedures* supplied by the verifiers may be general purpose procedures, not restricted to the context of the *rationality authority*. They should be able to check proofs [3] in an agreed upon format, a detailed logic proof in 3-SAT or Coq, probabilistically checkable proofs, interactive proofs in the style of zero-knowledge proofs, globally agreed upon, and simple instructions for the *agents* to check the proof (or even an empty proof relying on the verifier procedure to check the suggested actions in the style of nondeterministic Turing machines). The *verifiers* may use a library for the specification of the solution concepts and inform the user concerning the solution concept used and the consequences of the choice.

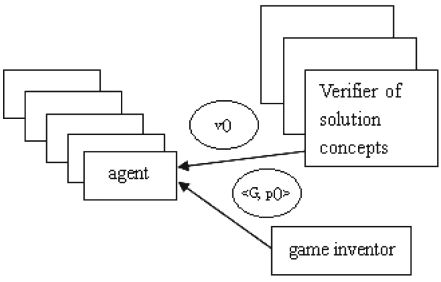

Fig. 1. The inventor sends the game G with a procedure $p()$ to the *agents*, suggesting actions and proofs for the optimality of the suggested actions. A designated verifier $v()$ sends *verification procedures* to the *agents* to allow verification of the proofs.

Verifying a best-reply could be as hard as computing it [24,29]. Computing the best-reply is known to be intractable [6]. However, there are some cases in which the game outcome is known, say, due to human innovation or statistically emerging patterns [14]. For example, taxation authorities or system administrators can sometimes observe the outcome and advise the participants about employing Nash equilibria, as in [32]. Thus, we may study the verification of

such advised solutions, rather than only their (possibly unrevealed way of) computation.

Online auctions have gained tremendous popularity in electronic commerce, B2B applications, and Internet ad auction applications [11]. Much of the literature models these auctions as single stage games or as repeated games. In some of our examples for the use of the *rationality authority*, we take a realistic approach in which the *agents* join the game in some random order, rather than participating in all game rounds (see Sects. 5 and 6).

Related Work

All agents are aware of the existence of the *rationality authority* as common knowledge. Since it communicates with agents before they choose their actions, one might view the authority as synchronization mechanisms that are used in correlated equilibria [1] or as moderators that are used in multi-party computation [15]. However, the *rationality authority* is *not* trusted, where as synchronization mechanisms are. Vis., the inventors must demonstrate their trustworthiness and have only the (trusted) verifiers at their disposal. This assumption is directly implied by the separation principles between the inventors and the verifiers.

Guerin and Pitt [19] present a framework for verification and compliance in a multi-*agent* system. They discuss whether the verification depends on the information that may be available (*agent* internals, observable behavior, normative specifications) and the semantic definition of the communication language. Moreover, they consider the types of languages that permit verification and testing in open systems where *agents*' internals are kept private. Their analysis is useful in enforcing compliance in open systems. Guerin [17] explains how to formally specify and verify subgame perfect equilibria. Tadjouddine [29] considers the complexity of verifying an equilibrium. Tadjouddine shows that Nash and Bayesian Nash equilibria can be verified in polynomial time. Moreover, dominant strategy equilibrium is NP-complete. Other related work includes [18,20,28,30,31], to name a few. None of the aforementioned results considers non-revealing verification methods. In [9,10] we proposed the *game authority*, which is a self-stabilizing middleware for assuring that all *agents* honestly follow the rules of the game. In [8,9] we presented a self-stabilizing mechanism for deterring joint deviations.

Our Contribution

We propose the *rationality authority* infrastructure that separates the interest, benefit and goals of the parties (*game inventors*, *verifiers* and *agents*) that enables *agents* to take rational (feasible and optimal) actions. The separation also includes the disjointment of the *game inventor* from game revenues and the *verifier* from selling reliable *verification procedures*. We propose the following specific case studies for the usefulness of the *rationality authority* framework.

– We explain how to use interactive theorem provers for verifying pure Nash equilibria (Sect. 3).

- We present an equilibrium verification method that does not reveal the *agent* preferences, and by that, preserves the users privacy and secures their actions when acting upon the advised equilibria. As a second case study, we present a general 2-*agents* (bimatrix) game for which a unique Nash equilibrium exists but it is hard to compute. For this game, we present a polynomial-time equilibria verifier with privacy guarantees. Namely, the verifier does not reveal the *agent* preferences in a way that resembles zero-knowledge proofs [16] (Sect. 4).
- We present the *participation* game, which has an equilibrium that is hard to compute without the *game inventor* advice. We show how to use the advice for computing the game equilibrium and verify it (Sect. 5).
- We study competitive on-line games in which each *agent* joins the game at a different time (as in [12]). The *game inventor* keeps statistical information about past *agent* actions. Each *agent*, upon arrival, has to choose a strategy. With probability p, the *agent* follows the inventor's suggested strategy. With probability $(1-p)$, it chooses a strategy based on its knowledge about the strategic (off-line) version of the game. The inventor chooses a strategy for the *agent* based on its statistics. When the inventor suggests a strategy, it must convince the *agent* that the strategy is beneficial for it. To do so, the inventor provides the *agent* with a formal proof that can be checked by a trusted verifier (Sect. 6).

The *rationality authority* enables *agents* to identify and take rational choices. Not only does the *rationality authority* verify the feasibility and optimality of proposed equilibria, but it can also cooperate with the *game authority* proposed in [9,10] that guarantees that the *agents* employ the strategy equilibrium by following the game rules.

This work appears as a brief announcement in [7].

2 Preliminaries

We use N to denote the set of *agents* that rationally and unilaterally choose an action from the set A_i, where $i \in N$ is an *agent*. A preference relation, $\succsim_i$, expresses the desire of the individual *agent* for one particular outcome over another. For game, G, the relation $\succsim_i$ refers to *agent* i's preferences. We can replace $\succsim_i$ with a *payoff/utility function* $u_i : A \to \mathbb{R}$, for which $u_i(a) \geq u_i(b)$ whenever $a \succsim_i b$. Namely, u_i is a function $A \to \mathbb{Z}$, which associates with the strategies of each *agent* the gain of *agent* i, where $A = A_0 \times \cdots \times A_{n-1}$. We represent single stage games, G, in their strategic form as $\langle N, A = (A_i), \succsim = (\succsim_i)\rangle$ (or $\langle N, A = (A_i)_{i\in N}, U = (u_i)_{i\in N}\rangle$).

Profiles

A strategy *profile* (a strategy combination) $a = (a_i)_{i\in N}$ is a strategy set, one for each *agent*, that completely implies all game actions by specifying a single strategy for each *agent*. We borrow from [27] also the notation of profiles that

Types

n; *number of agents*

TSi; *associate each agent with its number of strategies*

Si; *for each agent* $i, S_i(i)$ *is the strategy played by* i

$u; u(i,Si)$ (*utility*) *is* i*'s gain for the strategy profile* Si

Functions and Properties

$change(Si,si,i)$; *a strategy profile that is different from* Si *by the strategy* si *for agent* i. *Note that this function can build all the strategies needed for proving that a strategy profile is a Nash Equilibrium*

$isStrat(n,TSi,Si)$; *verifies the strategy profile*$'s$ *size*; $\forall i \leq n : Si(i) \leq TSi(i)$

$eqStrat(n,Si1,Si2)$; *checks equality of two strategy profiles*; $\forall i \leq n : Si1(i) = Si2(i)$

$leStrat(n,u,Si1,Si2)$; *checks incomparability of* $Si1$ *and* $Si2$; $\exists i,j : u_i(Si1) < u_i(Si2) \wedge u_j(Si2) < u_j(Si1)$

$leStrat(n,u,Si1,Si2)$; *checks* $Si1 \leq_u Si2$; $\forall i \leq n : u_i(Si1) \leq u_i(Si2)$

Pure Nash Equilibrium (**definitions**)

$isNash(n,u,Si,TSi)$; *verifies that* Si *is a Nash equilibrium*;

$isStrat(n,TSi,Si) \wedge \forall i \leq n, si \leq Si(i) : u_i(Si) \leq u_i(change(Si,si,i))$

$isMaxNash(n,u,Si,TSi)$; *checks* $isNash \wedge \forall$ *Nash equilibrium* $S'i : Si \leq_s S'i$

Verifying a Nash Equilibrium (**proof scheme**)

$allStrat; \forall Si : isStrat(n,TSi,Si) \rightarrow eqStrat(n,Si,Si_1) \vee eqStrat(n,Si,Si_2) \vee \ldots eqStrat(n,Si,Si_m)$, *where* $\{Si\_j\}_{j \leq m}$ *are possible strategies*

$allNash; \forall Si : isNash(n,u,Si,TSi) \rightarrow eqStrat(n,Si,NSi_1) \vee eqStrat(n,Si,NSi_2) \vee \ldots eqStrat(n,Si,NSi_m)$, *where* $\{NSi_j\}_{j \leq m}$ *are Nash equilibrium*

$NashMax; \forall Si : isNash(n,u,Si,TSi) \rightarrow leStrat(n,u,Si,NSi) \vee noComp(n,u,Si,NSi)$

Fig. 2. Definition of the game model and its equilibria. These definitions are used for verifying Pure Nash equilibria.

do not include the strategy of a single agent, i.e., (a_{-i}, a_i), as well as the profile of action sets, A_{-i}. We say that the strategy profile $s \in A$ is greater than $s' \in A$ if $\forall i \in N : u_i(s) \geq u_i(s')$. We denote $s \geq_u s'$.

Nash Equilibria

A strategy profile $s \in A$ is a pure Nash equilibrium of game $G = \langle N, A = (A_i), \succsim = (\succsim_i) \rangle$, if $\forall i \in N$, $\forall s'_i \in A_i$, $u_i(s) \geq u_i((s_{-i}, s'_i))$. We say that a pure Nash equilibrium (PNE) s is maximal if for any pure Nash equilibrium s', we do not have $s' \geq_u s$.[1] A game may not possess a PNE at all. However, if we extend the game to include *mixed strategy* by allowing each *agent* to choose its strategy with certain probabilities (and if we extend the payoff functions u_i to capture expectation), then an equilibrium is guaranteed to exist [23].

[1] Similarly, we can define the minimal Nash equilibria; s is minimal if for any pure Nash equilibrium s', we do not have $s' \leq_u s$.

3 Verifying a Nash Equilibrium Using Coq

Agents are expected to act rationally. We consider *agents* that are offered a strategy for which optimality is claimed. In order to familiarize the reader with equilibria verification, we briefly explain how to verify that a strategy profile, NSi, is a maximal Nash equilibrium. The proof presented in this section enumerates all strategy profiles, i.e., intractable with respect to both time and space for an unbounded number of *agents* or strategies (Sect. 4 addresses these important complexity issues). We sketch the proof of a pure Nash equilibrium using the theorem prover, Coq [2]. We use Coq because it is a standard and well-tested theorem prover.

The proof sketch considers the definitions that appears in Fig. 2. We note that Proposition *all_strat* (line 30) enumerates all strategy profiles. The proof of Proposition *all_strat* (line 30) is demonstrated by destructing all $Si(i)$ as long as $Si(i) \leq TSi(i)$, and then concluding that the equality exists with one of the strategies enumerated. The next step is to show that NSi is an equilibrium. Then, we enumerate all Nash equilibria (constructing a proposition for each of them). For each enumerated strategy in the Proposition *all_strat* (line 30), if it is an equilibrium, we use the corresponding proposition, if it is not, we show a counter-example (i and si such as $u(i, Si) < u(i, change(Si, si, i))$). NSi's optimality is showed by verifying that there is no equilibrium Si greater than the equilibrium NSi (Proposition $Nash\_max$, line 36). Proposition $Nash\_max$ assume that we are looking for a maximal equilibrium and in the opposite case we just have to change $le\_strat(n, u, Si, NSi)$ with $le\_strat(n, u, NSi, Si)$).

4 Provable Rationality Using Interactive Proofs

We study a 2-*agent* game, defined by the $n \times m$ matrices A, B of the payoffs of the two *agents*. Broadly speaking, the equilibrium is hard to compute. We present two interactive proofs that lead to an easy polynomial-time verification. The second proof also has privacy guarantees.

Case Study: A General 2-*agent* Game with Privacy Guarantees

We now turn to consider a 2-*agent* game, defined by the $n \times m$ matrices A, B of the payoffs of the two *agents* (the *row* agent , whose pure strategies are the n rows, and the column *agent*, whose strategies are the m columns). Here an equilibrium is, in general, hard to compute, i.e., complete in the complexity class PPAD, see [6]. However, the interactive proofs P_1 and P_2 (Figs. 3 and 4, respectively) lead to an easy polynomial-time verification with privacy guarantees in the case of P_2.

Lemma 1 shows that P_1's verifier algorithm has polynomial time complexity. The proof follows from the second Nash theorem [23], i.e., that for each strategy of the row *agent* in the support S_1 the expected gain should be the same and no less than the expected gain for strategies not in the support. It is easy to state the Verifier for the column *agent*.

Prover *(inventor)*: Provide each *agent* the *agents*' supports, i.e., strategy profiled played with non-zero probabilities.
Verifier of the row agent i: Let the support S_2 of the other *agent* (the column *agent*) be $\{j_1,\ldots,j_k\}$. Let $y_{j_1},\ldots,y_{j_k}$ be the Nash probabilities of the column *agent*. Let S_1 be the support of the row *agent* and $S_1=\{i_1,\ldots,i_\ell\}$. The verifier solves the linear system (1) and verifies that $0\le y_t\le 1$ for all $t\in\{j_1,\ldots,j_k\}$ and also that, for each row $i\notin S_1$, the expected gain $y_{j_1}A(i,j_1)+\cdots+y_{j_k}A(i,j_k)<\lambda_1$.

$$\begin{aligned}\lambda_1 &= y_{j_1}A(i_1,j_1)+y_{j_2}A(i_1,j_2)+\cdots+y_{j_k}A(i_1,j_k) \qquad (1)\\ &\vdots\\ \lambda_1 &= y_{j_\ell}A(i_\ell,j_1)+y_{j_2}A(i_\ell,j_2)+\cdots+y_{j_k}A(i_\ell,j_k)\\ y_{i_1}+\cdots+y_{j_k} &= 1\end{aligned}$$

Fig. 3. Interactive prover P_1.

Prover *(inventor)*: Send to each *agent* just *its* support, *its* probabilities, and the values λ_1,λ_2.
Verifier of the row agent i: Agent $i\in\{1,2\}$ asks the prover for two random indices j_1,j_2. If the prover is honest, it will return whether j_1 is in S_2 (or not) and whether j_2 is in S_2 (or not). Then the verifier computes the *expected gains of the other* agent for the two indices, $\lambda_2(j_1)$ and $\lambda_2(j_2)$. The verifier can then check whether
- "both j's in S_2", i.e., $\lambda_2(j_1)=\lambda_2(j_2)=\lambda_2$, and
- "1-in/1-out", say j_1 is in, i.e., $\lambda_2(j_1)=\lambda_2\ge\lambda_2(j_2)$.

The test is inconclusive for both $j_1,j_2\notin S_2$ but at least one will be in with probability at least $1/n$. Thus, on average, $O(n)$ random queries of the verifier will verify the equilibrium play.

Fig. 4. A private proof P_2.

Lemma 1. *The interactive proof P_1 has* verifier complexity *of time $LP(n,m)$ (where LP is the time of a linear program solver of at most n equations and m unknowns) and the number of bits communicated is $O(n+m)$.*

Proof Sketch. The verifier must solve a linear system of $k+1$ equations and $k+1$ unknown variables, where $k\ge\max(n,m)$ as implied by Fig. 3. Also, the prover just sends the two support sets (indices); each is less than $\max(n,m)$ in cardinality. Thus, it can actually send a vector of zeroes and ones, where the ones indicate the support indices. □

Note that our proof P_1 reveals both equilibrium supports to each *agent*. However, P_1 does not need to explicitly send any probability values. Moreover, the verifier algorithm P_2 extends P_1, still has a polynomial time, and yet does not reveal to any *agent* the Support (or probability values) of the other *agents*!

Remark 1. We can generalize the scheme of P_1 and P_2 to n *agents*. The prover provides the support sets $S_1,\ldots,S_n$ to all. The verifier of each *agent* then solves the corresponding *polynomial system* to find the Nash equilibrium probabilities.

Remark 2. The interactive proof P_2 does not reveal the actual equilibrium to either *agent*. Namely, the row *agent*, for example, cannot in general compute the Support (and hence the probability values) of the column *agent* if the row *agent* knows λ_1, λ_2 and its own Support and probabilities. To see this, consider the example in Fig. 5. Assume that the prover sends to the row *agent* its Support $S_1 = \{A\}$, its probabilities $p_A = 1, p_B = 0$, its payoff $\lambda_1 = 1$, and the payoff of the column player $\lambda_2 = 1$. Then, the row *agent* cannot conclude which is the actual equilibrium, since it is easy to see that any probabilities q_C, q_D of the column *agent* such that $q_C + q_D = 1$, $q \leq 1/2$, $q_C \geq 0$, $q_D \geq 0$ correspond to Nash equilibrium probabilities with $\lambda_2 = 1$.

Remark 3. In the case of large supports, e.g., $\theta(n)$, our verifier can test the equilibrium in a constant, k, number of queries, because the probability is constant in each case. Note that one can get the other's support via n queries, i.e., each query asks "is j in the other's support?" for all j to get all the support of the other *agents*. Thus, our verifier has a definite advantage in this case of large supports. The proposed test is always sublinear in n, except for the case of constant size supports.

	C	D
A	1, 1	1, 1
B	0, 1	2, 0

Fig. 5. A bimatrix game example.

5 Equilibrium Consultant with Provable Advices

We present the *Participation* game in which c is the auction participation fee and no gain is offered to the solo participant. The game's equilibrium is hard to compute without the *rationality authority*'s advice. We explain how the *agent* can use the advice for computing the game equilibrium and verify the *rationality authority*'s advice.

Consider n firms that are eligible to participate in an auction. The auction rules are:

- A firm f gets a value $v > 0$ if at least $k = 2$ firms choose to participate and f chooses *not* to.
- A firm f gets a value $v - c > 0$ when at least $k = 2$ firms participate and f is one of them.
- If nobody participates, then each firm gains zero.
- If firm f participates but the total number of participants is less than k, then f *pays* $c > 0$.

The Participation game is a symmetric game and thus, by Nash's theorem [23], it has a symmetric Nash equilibrium in which each firm decides to participate or not with probability p independent of the others.[2] The equilibrium stability implies equality between the expected payoffs of a firm f for participating and for not participating:

$$\begin{aligned}(v-c)\cdot \Pr\{\text{at least 1 other participates} \mid f \text{ participates}\} - \\ c\Pr\{\text{no other firm participates} \mid f \text{ participates}\} = \\ v\Pr\{\text{at least 2 other firms participate} \mid f \text{ does not}\} + \\ 0\cdot \Pr\{\text{at most 1 other firm participates} \mid f \text{ does not}\}\end{aligned} \tag{2}$$

Equation (2) defines the *equilibrium*'s probability p. Thus, the verifier can verify Eq. (2) by computing Eq. (3).

$$(v-c)\cdot A + (-c)\cdot B = v\cdot C + 0\cdot D, \tag{3}$$

where

$$\begin{aligned}A &= \Pr\{\text{at least 1 other firm participates} \mid f \text{ participates}\} = 1-(1-p)^{n-1}\\ B &= \Pr\{\text{no other firm participates} \mid f \text{ participates}\} = (1-p)^{n-1}\\ C &= \Pr\{\text{at least 2 other participate} \mid f \text{ does not}\} = 1-(1-p)^{n-1}-(n-1)p(1-p)^{n-2}\\ D &= \Pr\{\text{at most 1 other participates} \mid f \text{ does not}\} = (1-p)^{n-1}+(n-1)p(1-p)^{n-2}\end{aligned}$$

In fact, our simple example defines an easier job for the verifier since Eq. (3) gives Eq. (4).

$$\begin{aligned}(v-c)-(v-c)(1-p)^{n-1}+(-c)(1-p)^{n-1} &= v-v(1-p)^{n-1}-v(n-1)p(1-p)^{n-2}\\ (v-c)+(-v+c-c)(1-p)^{n-1} &= v-v(1-p)^{n-1}-v(n-1)p(1-p)^{n-2}\\ (v-c)-v(1-p)^{n-1} &= v-v(1-p)^{n-1}-v(n-1)p(1-p)^{n-2}\\ c &= v(n-1)p(1-p)^{n-2}\end{aligned} \tag{4}$$

For $\frac{c}{v}=\frac{3}{8}$, $n=3$, and $p=\frac{1}{4}$, the firm's expected gain is $v\left(1-\left(\frac{3}{4}\right)^2-2\cdot\frac{1}{4}\cdot\frac{3}{4}\right)=\frac{v}{16}$. In the case of any k, the prover still has to provide each firm with the equilibrium value of p and the verifier asserts Eq. (5).

$$\begin{aligned}(v-c)\cdot A_k + (-c)\cdot B_k = v\cdot C_k + 0\cdot D_k, \quad \text{where}\\ A_k = \Pr\{\text{at least } k \text{ firms participate} \mid f \text{ participates}\}\\ B_k = \Pr\{\text{at most } k-1 \text{ firms participate} \mid f \text{ participates}\}\\ C_k = \Pr\{\text{at least } k \text{ firms participate} \mid f \text{ does not}\}\\ D_k = \Pr\{\text{at most } k-1 \text{ firms participate} \mid f \text{ does not}\}\end{aligned} \tag{5}$$

Note that, now, p's value is hard to compute but, once it is given, it is easy to compute the conditional probabilities A_k, B_k, and C_k and verify the equilibrium

[2] We assume that firms do not differ significantly (or that they are aware of any difference among themselves) and thus a symmetric equilibrium, in which each firm participates with probability $p>0$, is natural to assume.

play in which a firm *expects to get the same* by using p to decide whether to play. Also note that for the Participation game, and for symmetric games in general, the players can cross-check that the prover has sent the *same* probability p to each of them, since it might be the case that more than one symmetric equilibrium exists. The existence of multiple equilibria would allow a dishonest prover to send different probabilities to the players, with each probability corresponding to a different symmetric equilibrium.

On-line Participation. Let us again assume that $k = 2$ and consider the case in which firms need to decide about their participation *at different times*. If firm f is the last to choose, the prover's "proof" is either $p = 1$, when at least one other firm has entered the game, or $p = 0$ otherwise. If the advice is $p = 1$, firm f will gain $v - c = \frac{5v}{8}$ and if $p = 0$, firm f will gain v. In both cases, f gains more in "on-line" advice in this setting. However, this verification method reveals the number of firms that have already played.

Of course, such advice favors the late arriving *agents*. But *if the order of arrivals is random*, the expected gain of any firm after advice is at least $\frac{1}{3} \cdot \frac{5v}{8} = \frac{5v}{24}$, still better than $\frac{v}{16}$ in the off-line case. On the other hand, false advice to the last *agent*, i.e., a flip of the value of p, will result in a loss! Thus it is crucial here to verify that the advice given by the prover is truthful. Namely, that it can lead to a best-reply given past history.

6 On-line Network Congestion Games

We study competitive games in a setting in which each *agent* joins the game at a different time (on-line games [12]). The *game inventor*, named the inventor, keeps statistical information about past *agents*. Each *agent*, upon arrival, has to choose a strategy. With probability p, the *agent* follows the inventor's suggested strategy. With probability $(1 - p)$, it chooses a strategy based on its knowledge about the strategic (off-line) version of the game. The inventor chooses a strategy for the *agent* based on its statistics. When the inventor suggests a strategy, it must convince the *agent* that the strategy is beneficial. To do so, we assume that the inventor provides the *agent* with a formal proof that can be checked by a trusted verifier (as in Sect. 5).

After defining an online variation of congestion games, such as [13], we present a greedy strategy for choosing a path based on the inventor's statistics. Let us consider a communication network, $N = (V, E, (d_e)_{e \in E})$, where V is the set of nodes, E is the set of arcs, and $d_e : \mathbb{R}_+ \to \mathbb{R}_+$ is a non-decreasing function for each $e \in E$, indicating the delay on arc e as a function of its congestion, i.e., the total load on it. Initially, the set of *agents* (the network users) is unknown to the inventor, which in the case of on-line congestion games is the operator of the network. We assume, however, that the number of *agents*, n, is known. Each *agent* i, at some point τ_i, joins the network and chooses a path π_i from a source node $s_i \in V$ to a sink node $t_i \in V$ to route its load $w_i \in \mathbb{R}_+$. The decision of each *agent* on the path is irrevocable. Let $[i] = \{1, \ldots, i\}$. The *configuration* of the network

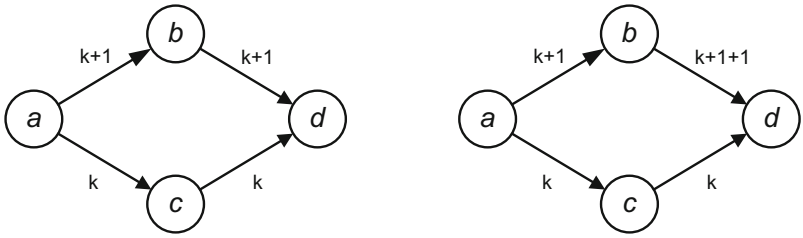

Fig. 6. An example in which the delay of each edge e is $d_e(x) = x$. Consider unit loads, and *agent* $2k+1$ that chooses a path from a to d. Observe that each edge has congestion k. A best-reply for *agent* $2k+1$ would be $a \to b \to d$ (shortest path). Suppose that the next *agent* to enter the network, *agent* $2k+2$, has to choose a path from b to d. Its only option is the path $b \to d$. Therefore, at time τ_{2k+2}, the delay experienced by *agent* $2k+1$ is $2k+3$, while its best-reply would be path $a \to c \to d$ with a total delay of $2k+2$.

at time τ_i (right after *agent* i joins) is $\pi(i) = (\pi_j)_{j\in[i]}$. Given a configuration $\pi(k)$, let $W_e(\pi(k)) = \sum_{j\in[k]:e\in\pi_j} w_j$ denote the total load on arc $e \in E$. At time τ_k, the total delay experienced by *agent* i is then $\lambda_i(\pi(k)) = \sum_{e\in\pi_i} d_e(\pi(k))$.

The goal of each *agent* is to choose a path, π_i, from s_i to t_i so that $\lambda_i(\pi(n))$ is minimized in N. However, an *agent* $i \in [n-1]$ cannot be aware of the final configuration $\pi(n)$. At time τ_i, its best-reply is to choose a shortest path from s_i to t_i, but this path cannot remain a best-reply for *agent* i at time τ_n when the game ends. To see this, consider the network shown in Fig. 6. The goal of the inventor is to minimize total congestion $\Lambda(\pi(n)) = \sum_{e\in E} d_e(\pi(n))$.

Choosing a Path Based on the Inventor's Statistics

The question now is, how should an *agent* choose its path since it is not aware of the final configuration. We let each *agent* i have two options: either to choose a shortest path given $\pi(i-1)$, or to ask the inventor for a suggested path.

What is the statistical information that the inventor maintains? We consider two cases: In the first case, the inventor has *prior* knowledge about the loads of the *agents*, knows for example that they are drawn from some particular probability distribution. In the second case, the inventor dynamically updates its information about the loads. That is, at each time τ_i, assuming that the total number of *agents* n is known, the inventor knows that loads $w_1, \ldots, w_i$ have appeared, and expects for example $(n-i)$ loads of expected value $\frac{\sum_{k=1}^{i} w_k}{i}$.

Greedy Strategies for Parallel Links

Assume that the network consists of a set $[m] = \{1, \ldots m\}$ of m parallel links from a source node s to a sink node t. What is the best-reply of *agent* i of load w_i that arrives at time τ_i? The best-reply is not necessarily the least loaded link at time τ_i, because *agent* i knows that the game has not ended, and expects $n-i$

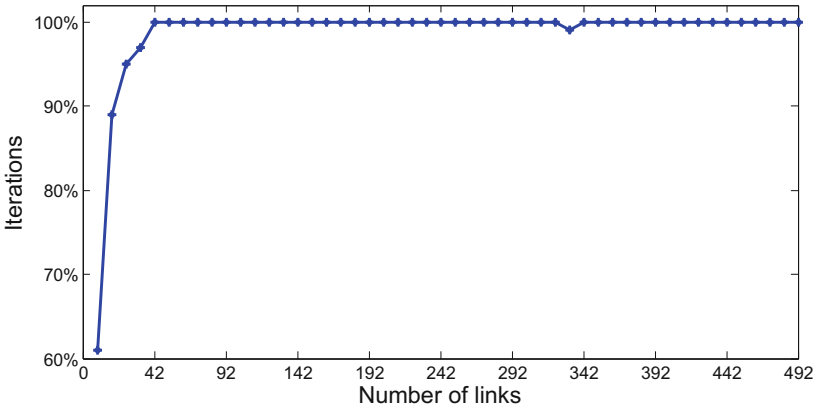

Fig. 7. The number of links (x-axis) and the iteration percentage (y-axis) in which the final assignment is strictly better, w.r.t. makespan, than the greedy strategy, see also Remark 4. We consider 1000 *agents*, uniform load distribution in $[0, 1000]$, the number of (equispeed) links is $m = 2, \ldots 500$.

loads to arrive. Namely, at time τ_i, *agent* i knows: (1) The total congestion on each link by time τ_i, (2) Its own load w_i, and (3) That $n - i$ loads are expected to arrive.

We simulate a simple on-line congestion game where *all agents* ask the inventor, i.e., $p = 1$ (see Fig. 7). We compare the greedy strategy (each *agent* on arrival chooses the least loaded link) to the strategy suggested by the inventor: The inventor takes into consideration the fact that more *agents* are expected. For each *agent* i, the inventor computes the average load $\overline{w_i}$ that has appeared so far.[3] Given the congestion on the links by time τ_i, agent i computes a Nash equilibrium assignment of its own load w_i and of $n - i$ loads $\overline{w_i}$. Namely, each load is assigned to the least loaded link, greatest load first. Then the inventor suggests that *agent* i choose the link that is suggested by that Nash equilibrium assignment. The greedy strategy is natural to assume while it also offers a performance guarantee (see Lemma 2); however, it is clear from the figure that it is outperformed by the strategy suggested by the inventor. The lemma refers to the term *makespan*, which is the maximum load on any link.

Lemma 2 (Greedy Strategy). *Let* $L^1, \ldots L^m$ *be the total loads of links* $1, \ldots m$ *when all* agents *have entered the game.* $L^j \leq \left(2 - \frac{1}{m}\right) \cdot OPT$*, where* OPT *is the optimum makespan (given all* w_i*'s).*

Proof. Let L_{ij} be the total load of link j right after *agent* i enters the game. Clearly, $L^j_n = L^j$. Let i_j be the last *agent* of load w_{i_j} that is assigned to link j.

[3] One can consider a way in which the agents can know that the inventor is not cheating about the average loads. For example, the system can require the inventor to publish the average loads with its signature at each round. In everyone record, then the inventor is kept responsible when found cheating, as in [10].

Since each *agent* chooses the least loaded link at the time it enters the game, Expression (6) holds for any link j.

$$\begin{aligned} L^j_{i_j-1} &\le L^k_{i_j-1} \quad \forall k \in [m] \setminus \{j\} & \qquad L^j_n - w_{i_j} &\le L^k_n \quad \forall k \in [m] \setminus \{j\} \\ L^j_{i_j} - w_{i_j} &\le L^k_{i_j} \quad \forall k \in [m] \setminus \{j\} & \qquad L^j - w_{i_j} &\le L^k \quad \forall k \in [m] \setminus \{j\} \end{aligned} \tag{6}$$

Expression (7) completes the proof by summing for all $k \in [m] \setminus \{j\}$.

$$(m-1)(L^j - w_{i_j}) \le \sum_{i=1}^{n} w_i - L^j \tag{7}$$

$$L^j \le \frac{\sum_{i=1}^{n} w_i}{m} + \frac{m-1}{m} w_{i_j} \le \frac{\sum_{i=1}^{n} w_i}{m} + \frac{m-1}{m} \max_i w_i \le \left(2 - \frac{1}{m}\right) \cdot OPT$$

Remark 4. Note that in Fig. 7, we plot the percentage of iterations where the strategy suggested by the inventor outperforms the greedy strategy. Each iteration involves an experiment, which considers random numbers, i.e., the agents' loads. The chart illustrates that, for sufficiently large number of links, obeying to the inventor's suggestion outperforms greediness in the vast majority of iterations. Note that we also observe particular cases in which the greedy strategy outperforms the inventor, e.g., in the experiment with 332 edges, the inventor was better at 99 % of the cases than the greedy strategy.

7 Discussions

This work studies the *rationality authority* infrastructure for encouraging computer *agents* to identify and make rational choices that are feasible and optimal. The *agents* can use this infrastructure for consulting with possibly biased *game inventors*. The *agents* verify these advices by using the *verification procedures*.

Anecdotes

There is a story about two folks, Ron (the rational) and Norton (the irrational), who walk in a far away road in the middle of a rainy night. At some point they both decide to sleep. Ron chooses to sleep on the muddy side of the road, in order to avoid cars that may drive in the paved part of the road. Norton decides to sleep on the more convenient paved part. A car arrives, the driver sees Norton at the last minute, and turns to the side of the road, exactly where Ron decided to sleep... Later, Norton may claim that he could not predict the influence of his irrational action on Ron. The existence of *rationality authority* suggests the way to act and produces a check-able proof for the optimality of the suggestion, eliminates the possible validity of Norton's excuse and may be used (after auditing Norton's actions) to blame Norton for not using the *rationality authority* results to act rationally.

The theory of non-cooperative games considers *agents* that are capable of identifying and making rational choices. However, non-cooperative game analysis is complicated; it is the subject of extensive theoretical study [27]. In certain

games the ingenious observations that are needed in order to figure the game outcome are even beyond the computer's capabilities; many solution concepts have no polynomial time decidability [26]. Such difficulties could be circumvented when the *game inventor* has additional capabilities that enables the *game inventor* to compute and propose solutions.

We consider *game inventors* that may have conflicts of interest with the *agents* and attempt to misadvise them. Therefore, we require the *game inventor* to equip the *agents* with a procedure to determine their actions for the game and with a procedure that produces a rationality proof of the chosen actions. The *agent* privately uses the procedures for choosing actions, possibly without revealing their preferences (utilities). The *agents* may suspect that the supplied procedures are biased or incorrect, as the inventor may benefit from the game. Thus, the outcome of the procedure that defines the actions and the outcome of the procedure that supplies the proof for the rationality of the chosen actions are checked using proof *verification procedures* (that are possibly provided by several verifiers).

The *rationality authority* design considers computational constraints as well as computer security considerations. The *game inventor* that suggests the actions and proofs, supplies procedures that are executed by the *agents* on their computers, where users execute the procedures, with their preferences (utilities), that are unknown to the rest of the world. To prevent information leakage, the agents may protect the activity in their computers by isolating them from the communication network. The *rationality authority* is designed to enable rational behavior of *agents*, whether they are humans or processes acting as part of electronic commerce.

Consider lottery with x raffle tickets to be sold. When the lottery is fair, the possibility to win, after buying a raffle ticket, is $1/x$. Suppose that the (*game inventor*, which is the) lottery company, knows that there are fake raffle tickets, which are almost indistinguishable from the valid ones. The lottery company knows that these fake tickets are being sold in a certain geographic area $\mathscr{A}$. The lottery company can advise the lottery participants to avoid buying tickets sold in area $\mathscr{A}$, supplying convincing proofs for identifying these fake raffles. By doing so, the lottery company allows the lottery participants to keep their chances at $1/x$. In this case, the information disclosure is minimal but very useful to the *agents*. The *rationality authority* can support such scenarios.

Conclusions

We focus on verification methods that do not violate the *agents*' privacy by revealing their preferences (utilities). Autonomous *agents* do not voluntarily reveal their preferences, because it could jeopardize the success of their actions. Moreover, even when such preferences are known to a trusted third party, security concerns and privacy restrictions limit the use of such information. This work presents examples in which such parties privately consult the *agents* using knowledge that only they have, as in [32], and yet offer proof for their advices, unlike [32]. Moreover, the local equilibrium verification allows us to consider a

more general scenario in which the *agents* have private and public preferences (as in [25]). Future research can further investigate efficient private verification of online games and online best replies [24].

Once the *rationality authority* requirements are satisfied, a *game authority* [9, 10] can guarantee that all *agents* take rational and honest actions; actions that follow the game rules. Moreover, actions of dishonest *game inventors*, *agents*, and *verifiers* can exclude the participant from acting in games and can be reported to a *reputation system* that audits their actions (e.g., see [8,9]).

The ordinary Joe and Jane do not have sufficient experience or the academic background for choosing best-replies and "perfect" maximum expected utility [21]. Interestingly, they are assured of making the right choice when using the *rationality authority*.

References

1. Aumann, R.J.: Subjectivity and correlation in randomized strategies. J. Math. Econ. **1**(1), 67–96 (1974)
2. Bertot, Y., Castéran, P., Huet, G., Paulin-Mohring, C., Pierre, C.: Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions. Springer, New York (2004)
3. Brânzei, S., Procaccia, A.D.: Verifiably truthful mechanisms. CoRR abs/1412.0056 (2014)
4. Burkhard, H.-D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.): CEEMAS 2007. LNCS (LNAI), vol. 4696, pp. 11–21. Springer, Heidelberg (2007)
5. Coy, P.: The secret to google's success. Business Week/Bloomberg L.P., 6 March 2006. (ts innovative auction system has ad revenues soaring)
6. Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a nash equilibrium. Commun. ACM **52**(2), 89–97 (2009)
7. Dolev, S., Panagopoulou, P.N., Rabiey, M., Schiller, E.M., Spirakis, P.G.: Brief announcement: Rationality authority for provable rational behavior. In: PODC 2011 TR 2011:03, Department CSE, Chalmers University of Technology (2011)
8. Dolev, S., Schiller, E.M., Spirakis, P.G., Tsigas, P.: Strategies for repeated games with subsystem takeovers implantable by deterministic and self-stabilizing automata. In: Manzalini, A. (ed.) Autonomics. ACM International Conference Proceeding Series, ACM (2008)
9. Dolev, S., Schiller, E.M., Spirakis, P.G., Tsigas, P.: Robust and scalable middleware for selfish-computer systems. Comput. Sci. Rev. **5**(1), 69–84 (2011)
10. Dolev, S., Schiller, E.M., Spirakis, P.G., Tsigas, P.: Game authority for robust and scalable distributed selfish-computer systems. Theor. Comput. Sci. **411**(26–28), 2459–2466 (2010)
11. Edelman, B., Ostrovsky, M., Schwarz, M.: Internet advertising and the generalized second-price auction: selling billions of dollars worth of keywords. Am. Econ. Rev. **97**(1), 242–259 (2007)
12. Foster, D.P., Vohra, R.: Regret in the on-line decision problem. Games Econ. Behav. **29**(1–2), 7–35 (1999)
13. Fotakis, D., Kontogiannis, S.C., Spirakis, P.G.: Atomic congestion games among coalitions. ACM Trans. Algorithms **4**(4), 52 (2008)

14. Freund, Y., Schapire, R.E.: Game theory, on-line prediction and boosting. In: COLT, pp. 325–332 (1996)
15. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, pp. 218–229. ACM, New York (1987)
16. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: Sedgewick, R. (ed.) Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6–8, 1985, pp. 291–304. ACM, Providence (1985)
17. Guerin, F.: An algorithmic approach to specifying and verifying subgame perfect equilibria. In: Proceedings of the Eighth Workshop on Game Theoretic and Decision Theoretic Agents (GTDT-2006), Hakodate, Japan (2006)
18. Guerin, F.: Applying game theory mechanisms in open agent systems with complete information. Auton. Agents Multi-Agent Syst. **15**(2), 109–146 (2007)
19. Guerin, F., Pitt, J.: Verification and compliance testing. In: Huget, M.-P. (ed.) Communication in Multiagent Systems. LNCS (LNAI), vol. 2650, pp. 98–112. Springer, Heidelberg (2003)
20. Guerin, F., Tadjouddine, E.M.: Realising common knowledge assumptions in agent auctions. In: IAT, pp. 579–586. IEEE Computer Society (2006)
21. Kahneman, D., Tversky, A.: Prospect theory: an analysis of decision under risk. Econometrica **47**(2), 263–291 (1979)
22. Mirrokni, V., Muthukrishnan, S., Nadav, U.: Quasi-proportional mechanisms: prior-free revenue maximization. In: López-Ortiz, A. (ed.) LATIN 2010. LNCS, vol. 6034, pp. 565–576. Springer, Heidelberg (2010)
23. Nash, J.F.: Equilibrium point in n-person games. Proc. Nat. Acad. Sci. USA **36**, 48–49 (1950)
24. Nikoletseas, S., Panagopoulou, P., Raptopoulos, C., Spirakis, P.G.: On the structure of equilibria in basic network formation. In: Gasieniec, L., Wolter, F. (eds.) FCT 2013. LNCS, vol. 8070, pp. 259–270. Springer, Heidelberg (2013)
25. Nisan, N., Ronen, A.: Algorithmic Mech. Des. **35**, 166–196 (2001)
26. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press, New York (2007)
27. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
28. Pauly, M.: Programming and verifying subgame-perfect mechanisms. J. Log. Comput. **15**(3), 295–316 (2005)
29. Tadjouddine, E.M.: Complexity of verifying game equilibria. CEEMAS **4**, 103–112 (2007)
30. Tadjouddine, E.M., Guerin, F.: Verifying dominant strategy equilibria in auctions. In: [4], pp. 288–297(2007)
31. Tadjouddine, E.M., Guerin, F., Vasconcelos, W.W.: Abstractions for model-checking game-theoretic properties of auctions. In: Padgham, L., Parkes, C.D., Müller, J., Parsons, S. (eds.) AAMAS (3), pp. 1613–1616. IFAAMAS, South Carolina (2008)
32. Thaler, R.H., Sunstein, C.R.: Nudge: Improving Decisions About Health, Wealth, and Happiness. Yale University Press, New Haven (2008)