Rev. Econ. Design (2008) 12:1–19
DOI 10.1007/s10058-008-0043-x

LEAD ARTICLE

Good neighbors are hard to find: computational complexity of network formation

Richard Baron · Jacques Durieu · Hans Haller · Rahul Savani · Philippe Solal

Received: 25 March 2005 / Accepted: 20 June 2007 / Published online: 11 March 2008

Abstract We investigate the computational complexity of several decision problems in a simple strategic game of network formation. We find that deciding if a player has a strategy that guarantees him a certain payoff against a given strategy profile of the other players is an **NP**-complete problem. Deciding if there exists a strategy profile that guarantees a certain aggregate payoff is also **NP**-complete. Deciding if there is a Nash equilibrium in pure strategies which guarantees a certain payoff to each player is **NP**-hard. The problem of deciding if a given strategy profile is a Nash equilibrium is investigated as well.

Keywords Strategic games · Network formation · Computational complexity

JEL Classification C72 · D02 · D85

We would like to thank both referees for valuable corrections and suggestions and one referee in particular for a very detailed and beneficial critique. We thank Colin de la Higuera for fruitful discussions. Support by the French Ministry for Youth, Education and Research, through project SCSHS-2004-04 is gratefully acknowledged.

R. Baron · J. Durieu · P. Solal
CREUSET, University of Saint-Etienne, 42023 Saint-Etienne, France

H. Haller (✉)
Department of Economics, Virginia Polytechnic Institute and State University,
Blacksburg, VA 24061-0316, USA
e-mail: haller@vt.edu

R. Savani
Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK

1 Introduction

A number of recent contributions in game theory have recognized that the organization of individuals into networks has a significant role in the determination of the aggregate behavior of many socio-economic systems. The emphasis has been on the understanding of how such networks form and evolve over time. Players are represented by nodes in the network, and their relationships by links between nodes. Within game theory, several strands of literature on network creation (network formation, network design) have emerged. For an introduction to this topic we refer to Dutta and Jackson (2003). The present paper deals with the computational complexity of certain yes–no decision problems for a class of simple network formation games.

Strategic network formation games. We consider network formation games based on three principles: (a) Links are formed simultaneously. (b) Players create links with other players by making the appropriate investment. (c) Players receive benefits from links to and from themselves. Such games can incorporate the idea that link formation is the result of a consent and that costs are two-sided (both players incur costs while forming a link). We restrict ourselves to the subclass of purely non-cooperative or strategic network formation games: Link formation and costs are one-sided while benefits can be one- or two-sided. A network is the outcome of a game in strategic form. The corresponding solution concept is Nash equilibrium. In general, the term Nash network refers to a network arising as the Nash equilibrium outcome of a network formation game. We shall use **Nash network** as a synonym for Nash equilibrium, since in our game (as in the literature in the tradition of Bala and Goyal 2000) a strategy profile can be identified with the resulting network.

Computation of Nash networks. Network formation games have applications in computer science (see Fabrikant et al. 2003; Anshelevich et al. 2003, 2004), industrial organization (see Belleflamme and Bloch 2004), sociology (see Whitmeyer 2002), and other fields.[1] For these applications, computation of Nash networks is an essential task. In general, computation of a Nash equilibrium in mixed strategies of a finite game poses a numerical challenge. A conceivable reason is the following: If there are at least six players and at least two pure strategies for each player, then determining the Nash equilibria in mixed strategies amounts to solving a system of multivariate polynomial equations of order five or higher which, as a rule, does not have an explicit solution. However, the lack of explicit solutions—or of rational solutions—should not pose a problem per se, provided that approximate solutions can be easily computed. Then the real issue becomes how easy or difficult it is to find a good approximation. It has recently been shown that finding one approximate equilibrium (ϵ-equilibrium in the sense of Radner (1980, 1981), to be precise) for a finite multi-player game in strategic form is complete for the complexity class PPAD.[2] This result suggests that it is unlikely that there exists a polynomial-time algorithm for finding one (approximate or exact) Nash equilibrium of such games.

[1] For potential applications in marketing, see Iacobucci (1996, 1998), Iacobucci and Hopkins (1992).

[2] See Daskalakis et al. (2005), Daskalakis and Papadimitriou (2005), and Chen and Deng (2005a,b). The complexity class PPAD was introduced in Papadimitriou (1994b).

Decision problems related to network formation games. The computational task seems to become less demanding, if one is content with searching for Nash equilibria in pure strategies. Since then it suffices to compare certain numbers given by the payoff functions (payoff vectors) of the game. Prima facie, the problem is further simplified, if one restricts oneself to the task of verifying whether a given pure strategy profile is a Nash equilibrium. We shall formulate this problem as a specific **decision problem**, NASH NETWORK VERIFICATION (NNV), for our class of network formation games: *Given a game and a pure strategy profile, is the strategy profile a Nash network of the game?* Solving the problem means to come up with an answer to the question, either "yes" or "no". It turns out that very likely, NNV is hard to solve. Before we get to study NNV in Sect. 4, we will focus on other basic decision problems whose investigation proves very useful for the analysis of NNV.

Input size and computing time. Polynomial running time of an algorithm refers to the asymptotic growth in computing time with respect to input size as the size of the problem (of the game) increases. Computing time refers to the worst-case running time. In the literature on computational complexity, polynomial growth of computing time is considered preferable to (more acceptable than, less complex than) non-polynomial or "exponential" growth. Indeed, for sufficiently large inputs, a problem with exponential computing time takes much longer than one with polynomial computing time. In particular, this difference becomes crucial when the limit of what is economically acceptable or feasible is reached. Suppose that for a given input size and computing technology, the limit of acceptable computing time or computing cost is reached. If computing becomes faster or cheaper, then the rate of growth of computing time determines whether a problem of larger input size can be handled in reasonable or affordable time.

Polynomial input complexity. For many interesting classes of games in strategic form, there exists a specific size parameter, a positive integer variable that determines the size of the game when all other parameters affecting size are kept constant. For example, it can be the number of pure strategies per player or the number of players. In our network formation game, the information needed to determine payoffs in an $(n + 1)$-player game with quadratic cost functions is given by $4n(n + 1)$ values for the benefit functions and $(n + 1)(n^2 + n + 1)$ coefficients for the cost functions, that is a vector of parameters whose dimension is of order n^3. Therefore, input size is polynomial in the number of players, even though each player has 2^n strategies. The game exhibits "polynomial input complexity" with respect to the number of players. This particular feature of our class of games is of utmost importance for the decision problems we consider: An algorithm that is polynomial in the number of players is also polynomial in the input size of the problem and vice versa.

Exponential input complexity. In contrast, in the class of network formation games with general cost functions, $(n+1)2^n$ parameters are needed to specify a cost function and input size is exponential in the number of players. More generally, the dimension of payoff vectors for an arbitrary $(n + 1)$-player game in strategic form where each player has $x > 1$ strategies is $(n + 1)x^{n+1}$, hence grows exponentially in n. Now consider the question whether an $(n + 1)$-player game with $x > 1$ strategies for each player and input size $k = (n + 1)x^{n+1}$ has a Nash equilibrium in pure strategies. This problem can be decided by making at most $k(x - 1) < k^2$ payoff comparisons. Hence computing time is polynomial in input size k and, therefore, the problem of deciding if a Nash equilibrium exists can be solved in polynomial time. However, while formally correct, the latter statement may be misleading: It ignores the fact that k grows exponentially in n and that the time it takes to input the data of the game is already exponential in n. Papadimitriou (2005) addresses this issue from the perspective of a leading expert on computational complexity: "But applying our field's methodology and norms to multiplayer games is not easy. Many computational problems related to multiplayer games (computing correlated equilibria, for instance) are "easy" in our way of thinking, but for the wrong reason: *The input length is exponential*—one number for each combination of strategies." Papadimitriou and Roughgarden (2005) refer to the "exponential input complexity of multi-player games."

Computational complexity of decision problems. In the sequel, we are considering decision problems like NNV which can be described by INSTANCES, the inputs, and a QUESTION, which has either "yes" or "no" as an answer. In NNV, an instance consists of a network formation game (the parameter values identifying the game, to be precise) and a pure strategy profile of that game and the question is if the strategy profile is a Nash equilibrium of the game. Each problem can be solved in finite time by a deterministic Turing machine, that is, given an instance of the problem, the machine will halt after finitely many steps and output either "yes" (be in the "yes" state) or "no" (be in the "no" state).[3]

Let **P** denote the class of decision problems that can be solved in polynomial time on a deterministic Turing machine. For many problems, it is unknown whether they belong to **P**. However, many among those problems have been shown to belong to the class **NP**.[4] Belonging to **NP** means that given a "yes" instance of the problem, there exists at least one polynomial-length certificate (string of symbols) so that in polynomial time, a deterministic Turing machine `accepts` the certificate as proof for a "yes" answer. Given a "no" instance of the problem, no certificate is ever accepted. A computational problem that in a sense (to be specified in Subsect. 2.3) is at least as hard as any other problem in **NP** is called **NP-hard**. A decision problem is **NP-complete** if it is both in **NP** and **NP**-hard.

Main results. Baron et al. (2006) show that deciding if there exists a connected Nash network that guarantees a given payoff to each player is an **NP**-hard problem, despite the simplicity of the network formation game. Here we look at the task of a single player to decide if there exist good or best responses against the others that guarantee a certain payoff to the player. Prima facie, this appears to be a less demanding task, since a standard method for computing Nash equilibria of a finite game reduces to an iterative myopic best-response procedure by the players. Yet our first main result, Theorem 3.1, suggests that computing a "good" response in a network formation game still is a computationally demanding task, since deciding if there exists a "good" response proves to be hard:

> *We obtain that deciding if there is a strategy for a player that guarantees him a certain payoff against the strategy profile of the other players is an* **NP***-complete problem.*

[3] For a definition in terms of formal languages, see Garey and Johnson (1979) and Papadimitriou (1994a).

[4] The most famous open question in theoretical computer science is $\mathbf{P} \overset{?}{=} \mathbf{NP}$—though it is widely believed that $\mathbf{P} \neq \mathbf{NP}$.

To interpret this result, imagine an individual who plans to connect to a network. That individual will certainly construct a procedure for determining the best (or sufficiently good) connections for all sets of characteristics of the network (benefits, costs) he can encounter. Unless $\mathbf{P} = \mathbf{NP}$, our first main result suggests the following: In some instances, the individual will be unable to come up with any procedure better than searching through all possible designs, that is all possible subsets of connections. The number of designs grows exponentially with the size of the network and, thus, possibly involves an exponential computing time. Consequently, from the point of view of this individual, good neighbors are hard to find.

Theorem 3.1 concerns the problem GOOD RESPONSE (GR) of deciding if there is a strategy for a player that guarantees him a certain payoff against the strategy profile of the other players. In Corollary 3.2, we deal with the problem BEST RESPONSE (BR) of deciding if there is a best response for a player that guarantees him a certain payoff against the strategy profile of the other players. In Corollary 3.3, we deal with the problem NASH NETWORK (NN) of deciding if there exists a Nash equilibrium that guarantees a certain payoff to every player. In Corollary 3.4, we consider the problem WELFARE of deciding if there exists a pure strategy profile that guarantees a certain aggregate payoff. GR, BR, and WELFARE are **NP**-complete and NN is **NP**-hard. More specifically, we are unable to provide a conclusive argument either for or against $\text{NN} \in \mathbf{NP}$. But we suggest in the proof of Corollary 3.3 that verifying the Nash network property may be a hard problem. This suggestion is an important reason for studying the problem NNV, which is also interesting in itself.

Our second main result, Theorem 4.3, deals with NASH NETWORK VERIFICATION (NNV), the problem of deciding if a given pure strategy profile is a Nash equilibrium for a given game. We find that NNV does not belong to the class **NP**, unless $\mathbf{NP} = \mathbf{co-NP}$, where the class $\mathbf{co-NP}$ consists of the decision problems such that for every “no” instance of a problem, there exists at least one polynomial-length certificate (string of symbols) so that in polynomial time, a deterministic Turing machine `accepts` the certificate as proof for a “no” answer. For a formal definition of $\mathbf{co-NP}$, we refer to Subsect. 2.3. Our second main result suggests: In a worst-case scenario, one has to consider a very large number of deviations—possibly every conceivable deviation of every player—in order to check if the given pure strategy profile constitutes a Nash equilibrium.

Outline. In Sect. 2, we present the strategic game of network formation and give a brief introduction to the theory of computational complexity. In Sect. 3, the computational complexity of several decision problems pertaining to the network formation game is investigated. Sect. 4 is devoted to the problem NNV. In Sect. 5, we make some qualifying remarks.

2 Preliminaries

From a computational perspective, it seems unlikely that players or analysts use information that grows exponentially with the number of players. As we have observed in the introduction, using that much information can lead to algorithms that are good in the classical sense, being polynomial in the input size of the problem, but only because the input is exponential, which could be considered unreasonably large. Therefore, it is natural to study the computational complexity of finding best responses and Nash equilibria in games with polynomial input complexity. While arbitrary network formation games can exhibit exponential input complexity, the game introduced in Subsect. 2.2 and investigated in the sequel has polynomial input complexity.

2.1 Games in normal form

The basic concept is a finite strategic game in normal form

$$\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}),$$

with $n+1$ players. $I = \{0, 1, \ldots, n\}$ denotes the finite set of players. For $i \in I$, S_i is i's strategy set, with generic element s_i, and $\mathcal{S} = \prod_{i\in I} S_i$ is the set of strategy profiles of all players, with generic elements $s = (s_i)_{i\in I}$. $S_{-i} = \prod_{j\neq i} S_j$ denotes the set of strategy profiles of all players except player $i \in I$, with generic elements $s_{-i} = (s_j)_{j\neq i}$. In slight abuse of notation, we sometimes write $s = (s_i, s_{-i})$. Player $i \in I$ has payoff function $u_i : \mathcal{S} \to \mathbb{R}$. For $i \in I$, let $Br_i : S_{-i} \twoheadrightarrow S_i$ be i's pure **best-response correspondence** which maps each strategy profile $s_{-i} \in S_{-i}$ to the non-empty and finite set

$$Br_i(s_{-i}) = \arg\max_{s_i\in S_i} u_i(s_i, s_{-i}).$$

The combined pure best-response correspondence $Br : \mathcal{S} \twoheadrightarrow \mathcal{S}$ of the game $\mathcal{G}$ is defined as the product set of all players' pure best-response correspondences,

$$Br(s) = \prod_{i\in I} Br_i(s_{-i}).$$

A **Nash equilibrium** in pure strategies is a fixed point of Br, that is, a strategy $s^* \in \mathcal{S}$ such that $s^* \in Br(s^*)$.

2.2 The network formation game

We model a network as the outcome of a strategic game in normal form. The finite set of nodes of the network is exogenously given and coincides with the set of players. A strategy profile in the game determines the directed links of the network. Each player's strategy is a vector $s_i = (s_{ij})_{j\neq i}$ where $i \in I$ and $s_{ij} \in \{0, 1\}$ for each $j \neq i$. The value $s_{ij} = 1$ means that player i and j have a link initiated by i whereas $s_{ij} = 0$ means that i does not initiate a link with j. A strategy profile $s = (s_i)_{i\in I} \in \mathcal{S}$ is equivalent to a digraph or a network. Within a network, benefits for the players are generated depending on how they are connected to each other. Here we will restrict our attention to **link-based benefits** or **dyadic benefits** in the sense of Chakrabarti and Gilles (2006) and Baron et al. (2006), respectively. More precisely, for each pair of players $(i, j) \in I \times I$, $i \neq j$, player i receives a benefit $b_{ij}(s_{ij}, s_{ji}) \in \mathbb{R}$ from interacting with j, if i chooses $s_{ij} \in \{0, 1\}$ and j chooses $s_{ji} \in \{0, 1\}$. Player $i \in I$ incurs a **cost** $c_i(s_i) \geq 0$ when choosing $s_i \in S_i$. The function c_i is strictly increasing in the number of links formed. Player i's payoff or net benefit from the network $s \in \mathcal{S}$ is given by the following expression:

$$u_i(s) = \sum_{j \neq i} b_{ij}(s_{ij}, s_{ji}) - c_i(s_i).$$

Quadratic cost functions. We assume that costs are quadratic, that is for $i \in I$, $s_i \in S_i$:

$$c_i(s_i) = c_0^i + \sum_{j \neq i} c_j^i s_{ij} + \sum_{j \neq i} \sum_{k \neq i} c_{jk}^i s_{ij} s_{ik},$$

with a vector of coefficients $(c_0^i, (c_j^i)_{j \neq i}, (c_{jk}^i)_{j \neq i; k \neq i}) \in \mathbb{R}_+^{n^2+n+1}$. Notice that $s_{ij} = s_{ij}^2$ for $s_{ij} \in \{0, 1\}$ and, therefore, the linear terms can be omitted.

This completes the basic description of the network formation game $\mathcal{G}$. Nash equilibria of $\mathcal{G}$ are also called **Nash networks**. Finally, we introduce a specific welfare measure as the sum of the payoffs of all the agents. Let $W : \mathcal{S} \to \mathbb{R}$ be defined as

$$W(s) = \sum_{i=1}^{n} u_i(s). \tag{1}$$

A network s is **efficient** if $W(s) \geq W(s')$ for all $s' \in \mathcal{S}$. An efficient network is necessarily Pareto-optimal. That is, there is no other network that provides a higher payoff to some player(s) and no lower payoff to any player. But not every Pareto-optimal network has to be efficient. Efficiency is a major performance criterion for network designers or planners and plays a prominent role in the traditional network literature. It is most attractive when payoffs are monetary and side-payments between players are feasible. Efficiency constitutes an important benchmark for network performance even when network formation is decentralized and structured as a strategic game.

2.3 NP-completeness

For a comprehensive introduction to **NP**-completeness the reader is referred to Garey and Johnson (1979) or Papadimitriou (1994a). Let **P** denote the class of decision problems that can be solved on a deterministic Turing machine by a polynomial-time algorithm, that is, polynomial in the length of inputs for an instance of the problem. Let **NP** denote the class of all decision problems which can be solved in polynomial time by means of a nondeterministic Turing machine. Instead of using the notion of nondeterminism, one can define the class **NP** in terms of the concept of **polynomial-time verification**. Roughly speaking, a verification algorithm is a deterministic algorithm which takes as input an instance of the problem and a string of symbols called a certificate. A decision problem belongs to **NP**, if for each "yes" instance of the problem, there exists at least one certificate (string of symbols) of polynomial length so that in polynomial time, the algorithm `accepts` the certificate as proof for a "yes" answer; in case of a "no" instance of the problem, no certificate gets accepted. When the question at hand is whether an object with a certain property exists, some certificates may represent candidate objects and the algorithm verifies in polynomial time whether such a certificate has the desired property.[5] Thus the class **NP** is the class of decision problems where "yes" instances can be verified in polynomial time.

The fundamental open question in computational complexity is whether $\mathbf{P} = \mathbf{NP}$. By definition $\mathbf{P} \subseteq \mathbf{NP}$. It is not known, however, whether all problems in **NP** can, in fact, be solved in polynomial time by a deterministic Turing machine. The generally accepted belief is that $\mathbf{P} \neq \mathbf{NP}$. In an effort to determine whether $\mathbf{P} = \mathbf{NP}$, the class of **NP**-complete problems has been introduced. We say that a problem P_1 is **polynomial-time reducible** to a problem P_2, written $P_1 \preceq_p P_2$, if

(i) There exists a function f which maps any instance of P_1 to an instance of P_2 in such a way that I_1 is a "yes" instance of P_1 if and only $f(I_1)$ is a "yes" instance of P_2.
(ii) For any instance I_1, the instance $f(I_1)$ can be constructed in polynomial time.

If P_1 is polynomial-time reducible to P_2, we can say that any algorithm for solving P_2 can be used to solve P_1. Intuitively, problem P_1 is "no harder" to solve than problem P_2. A problem P is said to be **NP-complete** if (i) $P \in \mathbf{NP}$, and (ii) for *every* problem $P' \in \mathbf{NP}$, $P' \preceq_p P$. If a problem satisfies condition (ii) but not necessarily condition (i), then we say that it is **NP-hard**. Let **NPc** denote the class of **NP**-complete problems.

The binary relation $\preceq_p$ is transitive on the set of decision problems. Because of this, a method frequently used in demonstrating that a given problem is **NP**-complete is the following:

(i) Show that $P \in \mathbf{NP}$, and
(ii) Show that there exists a problem $P' \in \mathbf{NPc}$ such that $P' \preceq_p P$.

It follows from the definition of **NP**-completeness that if any problem in **NPc** can be solved in polynomial time, then every problem in **NPc** can be solved in polynomial time, and $\mathbf{P} = \mathbf{NP}$. On the other hand, if there is some problem in **NPc** that cannot be solved in polynomial time, then no problem in **NPc** can be solved in polynomial time.

To a decision problem P, we can associate its complement P^c defined by the property that the "no" instances of P are the "yes" instances of P^c and the "yes" instances of P are the "no" instances of P^c. This allows to define classes of decision problems via properties of complements, in particular $\mathbf{co\text{-}NP} = \{P \mid P^c \in \mathbf{NP}\}$. Whether $\mathbf{NP} = \mathbf{co\text{-}NP}$ holds is another important open question, though it is widely believed that $\mathbf{NP} \neq \mathbf{co\text{-}NP}$. However, $\mathbf{P} = \mathbf{co\text{-}P}$, where $\mathbf{co\text{-}P} = \{P \mid P^c \in \mathbf{P}\}$. Therefore, $\mathbf{P} = \mathbf{NP}$ would imply $\mathbf{NP} = \mathbf{co\text{-}NP}$.

A key result, which will prove very useful later on, is

Lemma 2.1 *If* $\mathbf{NP} \neq \mathbf{co\text{-}NP}$ *and* $P \in \mathbf{NPc}$*, then* $P \notin \mathbf{co\text{-}NP}$.

[5] See the qualification following the proof of Corollary 3.2.

Proof See Proposition 4.10.2 in Papadimitriou (1994a) and Theorem 7.2 in Garey and Johnson (1979). □

There are many methods for computing one Nash equilibrium, called a "sample equilibrium" in a finite game. For a review of these methods, we refer to McKelvey and McLennan (1996). The computational complexity of deciding if a strategic game has a Nash equilibrium in pure strategies has been addressed first by Sahni (1974). Inspection of Sahni's proof shows that the problem is **NP**-complete for a class of finite games where each player has two strategies. For results with a fixed number of players and a variable number of individual strategies, we refer to Conitzer and Sandholm (2003), Gilboa and Zemel (1989), Koller and Megiddo (1992), Koller et al. (1996).

3 Computational complexity of networking

An adaptive procedure frequently used in evolutionary game theory generates a sequence of strategy profiles. At each iteration either all the players (as in Berninghaus and Schwalbe 1996a,b) or one select player (as in Blume 1995) compute a best response against the last play of the other players. That is players update their strategy—in a simultaneous or in an asynchronous way—by computing a myopic best response. Similarly, Topkis (1998) proposes algorithms to approximate Nash equilibria of supermodular games, where at each iteration, all players' or one player's strategies are updated by computing a best response against the last play of the other players. Instead of assessing the computing time for the entire procedure, we may ask how long it takes to perform one step of the procedure: How long would it take a player to determine a best response?

In this section we obtain results which are rather pessimistic in that regard for the class of network formation games $\mathcal{G}$.[6] We formulate the problem as a decision problem, i.e., a problem that has only two possible solutions, either the answer "yes" or the answer "no". Theorem 3.1 is about the problem of deciding if there exists a sufficiently good response. Corollary 3.2 is about the problem of deciding if there exists a sufficiently good best response. Both problems are **NP**-complete. Two additional corollaries deal with the problem of deciding if a Nash network exists that guarantees every player a certain payoff and with the problem of deciding if a network with sufficiently high aggregate payoff exists. The first problem proves **NP**-hard. The second is **NP**-complete.

Next we introduce all the decision problems we are going to consider in this section. Each instance of these problems is supposed to be finitely encoded.
GOOD RESPONSE (GR)
INSTANCE: A network formation game $\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I})$ of $n + 1$ players, a player $j \in I$, a joint strategy $t_{-j} \in S_{-j}$ of the other players, and a rational number B.
QUESTION: Is there a strategy $s_j \in S_j$ such that $u_j(s_j, t_{-j}) \geq B$?

[6] In Sect. 4, we will address another fundamental problem: to decide if a pure strategy profile is a Nash network.

BEST RESPONSE (BR)
INSTANCE: A network formation game $\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I})$ of $n+1$ players, a player $j \in I$, a joint strategy $t_{-j} \in S_{-j}$ of the other players, and a rational number B.
QUESTION: Is there a best response $s_j \in S_j$ against t_{-j} such that $u_j(s_j, t_{-j}) \geq B$?

NASH NETWORK (NN)
INSTANCE: A network formation game $\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I})$ of $n+1$ players and a rational number B.
QUESTION: Is there a Nash network $s = (s_i)_{i\in I} \in \mathcal{S}$ such that $u_i(s) \geq B$ for all $i \in I$?

WELFARE
INSTANCE: A network formation game $\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I})$ of $n+1$ players and a rational number B.
QUESTION: Is there a network $s = (s_i)_{i\in I} \in \mathcal{S}$ such that $W(s) \geq B$?

SUBSET SUM
INSTANCE: Positive integers $a_1, \dots, a_n$ and Z.
QUESTION: Does there exist a set of indices $A \subseteq \{1, \dots, n\}$ such that $\sum_{i\in A} a_i = Z$?

Theorem 3.1 GR *is* **NP**-*complete.*

Proof Suppose a non-deterministic Turing machine guesses an arbitrary strategy $s_j \in S_j$. In order to verify whether the certificate s_j gives a net benefit of at least B against t_{-j}, the machine has to take into account n dyadic interactions when computing player j's payoff—which demonstrates that GR is in **NP**.

Now we construct a reduction from a known **NP**-complete problem to GR. We use the SUBSET SUM problem (see Garey and Johnson 1979, p. 223). Given an instance $X = \{a_1, \dots, a_n, Z\}$ of the SUBSET SUM problem, we construct an instance $f(X)$ of GR as follows. First, set $I = \{0, 1, \dots, n\}$ and $j = 0$. Second, put $0 = b_{0i}(0,0) < b_{0i}(0,1) = b_{0i}(1,1) = b_{0i}(1,0) = 2Za_i$ for $i = 1, \dots, n$. This condition together with the specification of 0's cost function below implies that if $s_{i0} = 1$ then $s_{0i} = 0$. The argument runs as follows. If i has a link with player 0, then by definition of u_0, player 0 incurs an additional cost from setting $s_{0i} = 1$ as well, while he does not get any benefit from it. Third, for player 0 define the cost function as follows:

$$c_0(s_0) = \sum_{i=1}^{n}\sum_{k=1}^{n} a_i a_k s_{0i} s_{0k}.$$

Fourth, set $B = Z^2$. These operations can be computed in quadratic time. As a further operation, set $b_{ij} \equiv 0$ and let the cost function c_i be of the special linear form $c_i(s_i) = \gamma_i \sum_{k\neq i} s_{ik}$ for $i = 1, \dots, n$, and $j \neq i$, which are computable in polynomial time. As a last operation, set $t_i = (0, \dots, 0)$ for $i = 1, \dots, n$. It follows that an instance $f(X)$ of GR can be constructed in polynomial time. Denote $f(X) = [\mathcal{G}, 0, t_{-0}, B]$.

We claim that there exists $s_0 \in S_0$ with $u_0(s_0, t_{-0}) \geq Z^2$ if and only if there exists $A \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in A} a_i = Z$. First, consider $s_0 \in S_0$. We have

$$\begin{aligned} u_0(s_0, t_{-0}) &= 2Z \sum_{i=1}^{n} a_i s_{0i} - \sum_{i=1}^{n} \sum_{k=1}^{n} a_i a_k s_{0i} s_{0k} \\ &= -\left(\sum_{i=1}^{n} a_i s_{0i} - Z \right)^2 + Z^2 \end{aligned} \tag{2}$$

Now suppose that there exists $A \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in A} a_i = Z$. Construct the strategy $s_0 \in S_0$ as follows: $s_{0i} = 1$ for $i \in A$ and $s_{0i} = 0$ for $i \notin A$. Then $u_0(s_0, t_{-0}) = Z^2$, the best net benefit player 0 can expect against t_{-0}. If there does not exist such a subset A, then $u_0(s_0, t_{-0}) < Z^2$ for each strategy s_0 in S_0, which proves the Theorem. □

In the proof, any quadratic cost function c_i for $i \neq 0$ will do. The restriction to the special linear form $c_i(s_i) = \gamma_i \sum_{k \neq i} s_{ik}$ is used in the proofs of Corollaries 3.3 and 3.4.

Corollary 3.2 BR *is* **NP**-*complete.*

Proof In the foregoing reduction from SUBSET SUM to GR, $u_0(s_0, t_{-0}) = Z^2$ is the best net benefit player 0 can expect against t_{-0}. Thus, we have also constructed a reduction from SUBSET SUM to BR.

Given an instance $[\mathcal{G} = (I, (S_i)_{i \in I}, (u_i)_{i \in I}), j, t_{-j}, B]$ of BR and a certificate $s_j \in S_j$, checking for $u_j(s_j, t_{-j}) \geq B$ will do, since existence of any strategy t_j with $u_j(t_j, t_{-j}) \geq B$ is equivalent to existence of a best response s_j with $u_j(s_j, t_{-j}) \geq B$. This equivalence holds because of the finiteness of S_j. □

Notice that in the second part of the proof, an algorithm need not verify both that the certificate s_j satisfies $u_j(s_j, t_{-j}) \geq B$ and that it is a best response. The only requirement is that at least one certificate is accepted in case of a "yes" instance of BR and no certificate is `accepted` in case of a "no" instance of BR. See Garey and Johnson (1979, p. 28) and Papadimitriou (1994a, p. 46).

Further notice that BR is in **P** if costs are linear. For suppose that for each $i \in I$, there exist $\gamma_{ik} > 0$, $k \in I \setminus \{i\}$, such that

$$c_i(s_i) = \sum_{k \neq i} \gamma_{ik} s_{ik}.$$

In this case, the $(n+1)$-player network formation game can be decomposed into dyadic components. The net benefit of a player i is the sum of the terms $b_{ik}(s_{ik}, s_{ki}) - \gamma_{ik} s_{ik}$ over $k \neq i$. From the point of view of player j, the computation of a best response against $t_{-j} \in S_{-j}$ can be decomposed into n independent (local) best responses: For each dyad (j, k), $k \neq j$, player j has two alternatives, $s_{jk} = 0$ or $s_{jk} = 1$. Clearly each of these computations can be carried out in polynomial time. Thus for the special case of linear costs, BR is in **P**.

Corollary 3.3 NN *is* **NP**-hard.

Proof The proof of Theorem 3.1 and Corollary 3.2 shows that BR remains **NP**-complete when restricted to instances $[\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ where player j has a quadratic cost function and all other players have linear cost functions of the form $c_i(s_i) = \gamma_i \sum_{k\neq i} s_{ik}$. Denote this problem BR*.

Next we construct a reduction from BR* to NN. Given an instance $[\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ of BR*, we construct an instance $[\mathcal{G}' = (I', (S'_i)_{i\in I'}, (u'_i)_{i\in I'}), B']$ of NN as follows.

Set $I' = I$ and $B' = B$. For $i \neq j, k \neq i$, $(s_{ik}, s_{ki}) \in \{0, 1\}^2$, set $b'_{ik}(s_{ik}, s_{ki}) = B/n + \gamma_i t_{ik}$ if $s_{ik} = t_{ik}$, $b'_{ik}(s_{ik}, s_{ki}) = (B-1)/n$ if $s_{ik} \neq t_{ik}$, and $c'_i = c_i$. For $k \neq j$, $(s_{jk}, s_{kj}) \in \{0, 1\}^2$, set $b'_{jk}(s_{jk}, s_{kj}) = b_{jk}(s_{jk}, s_{kj})$. Set $c'_j = c_j$.

Then for each $i \neq j$, t_i is a strictly dominant strategy for i in $\mathcal{G}'$ which yields payoff B to i. Therefore, $s_j \in S_j$ is a best response against t_{-j} in $\mathcal{G}$, with payoff $u_j(s_j, t_{-j}) \geq B$ if and only if (s_j, t_{-j}) is a Nash network of $\mathcal{G}'$ with $u'_i(s_j, t_{-j}) \geq B$ for all $i \in I$.

This shows that $[\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ is a "yes" instance of BR* if and only if $[\mathcal{G}' = (I', (S'_i)_{i\in I'}, (u'_i)_{i\in I'}), B']$ is a "yes" instance of NN. We had to define $4n(n+1)$ numbers $b_{ik}(s_{ik}, s_{ki})$. Since $j's$ cost function is quadratic and all the other cost functions for $\mathcal{G}'$ are linear, the cost parameters can be constructed in quadratic time. Hence the parameters of the instance $[\mathcal{G}' = (I', (S'_i)_{i\in I'}, (u'_i)_{i\in I'}), B']$ can be obtained from those of the given instance $[\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ in polynomial time.

One may have to make payoff comparisons with all the other $2^n - 1$ strategies of player j in order to verify whether s_j is a best response against t_{-j}. Therefore, we cannot conclude that NN belongs to **NP**. Hence NN is **NP**-hard, but has not been shown to be **NP**-complete. $\square$

The open question whether NN belongs to **NP** leads us to study the problem NNV in Sect. 4. Problem NNV of deciding if a given pure strategy profile is a Nash equilibrium (Nash network) for a given game is also interesting in itself.

Corollary 3.4 WELFARE *is* **NP**-*complete.*

Proof Given an instance $[\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}),\ B]$ of WELFARE and a certificate $s = (s_i)_{i\in I} \in \mathcal{S}$, it takes polynomial time to compute $W(s)$ and to verify whether $W(s) \geq B$. Hence WELFARE belongs to **NP**.

The proof of Theorem 3.1 shows that GR remains **NP**-complete when restricted to instances $[\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ where player j has a quadratic cost function and all other players have linear cost functions of the form $c_i(s_i) = \gamma_i \sum_{k\neq i} s_{ik}$. Denote this problem GR*. Next we construct a reduction from GR* to WELFARE. Consider an instance $[\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ of GR*. We construct an instance $[\mathcal{G}' = (I', (S'_i)_{i\in I'}, (u'_i)_{i\in I'}), B']$ of WELFARE as follows.

Set $I' = I$ and $B' = B$. For $i \neq j, k \neq i$, $(s_{ik}, s_{ki}) \in \{0, 1\}^2$, set $b'_{ik}(s_{ik}, s_{ki}) = \gamma_i s_{ik}$ and $c'_i = c_i$. For $k \neq j$, $(s_{jk}, s_{kj}) \in \{0, 1\}^2$, set $b'_{jk}(s_{jk}, s_{kj}) = b_{jk}(s_{jk}, t_{kj})$. Set $c'_j = c_j$.

Then $W'(s_j, s_{-j}) = u_j(s_j, t_{-j})$ for all $(s_j, s_{-j}) \in S_j \times S_{-j}$. Hence $W'(s_j, s_{-j}) \geq B$ implies $u_j(s_j, t_{-j}) \geq B$ and $u_j(s_j, t_{-j}) \geq B$ implies $W'(s_j, t_{-j}) \geq B$. Consequently, we find that $[\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ is a "yes" instance of GR* if and only if $[\mathcal{G}' = (I', (S_i')_{i\in I'}, (u_i')_{i\in I'}), B']$ is a "yes" instance of WELFARE. The benefit and cost parameters of the instance $[\mathcal{G}' = (I', (S_i')_{i\in I'}, (u_i')_{i\in I'}), B']$ can be obtained from those of the given instance $[\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ in polynomial time. □

There exists an alternative derivation of Corollary 3.4 via a direct reduction from SUBSET SUM to WELFARE, which is very similar to the proof of Theorem 3.1.

Since for a finite game $\mathcal{G}$, a welfare maximizer always exists, the following two questions are logically equivalent for an instance $[\mathcal{G}, B]$ of WELFARE:

QUESTION 1: Is there a network $s \in \mathcal{S}$ such that $W(s) \geq B$?

QUESTION 2: Is there a network $s \in \mathcal{S}$ such that $W(s) \geq B$ and $W(s)$ is maximal?

In particular, if there exists s with $W(s) \geq B$, then there exists an efficient s', possibly different from s, with $W(s') \geq B$, which yields the logical equivalence. Let us call the problem associated with the second question EFFICIENT NETWORK (EN). Then a "yes" instance $[\mathcal{G}, B]$ of WELFARE is a "yes" instance of EN and vice versa. Consequently, the reduction in the proof of Corollary 3.4 also works for EN instead of WELFARE. Moreover, similar to the argument in the proof of Corollary 1, verification of WELFARE is equivalent to verification of EN. Therefore, EN is **NP**-complete as well.

4 Nash network verification

In this section we address the problem of deciding if a pure strategy profile is a Nash network of the network formation game. This constitutes an interesting question on its own. It is also related to an unanswered question in the previous section. Specifically, we were unable to conclude that NN belongs to **NP**, because we could not show that given a game $\mathcal{G}$ and a strategy profile $s \in \mathcal{S}$, it can be verified in polynomial time that each component s_j is a best response against s_{-j}. In other words, we could not show that it can be verified in polynomial time that s is a Nash network of $\mathcal{G}$. Nor could we prove that it cannot be verified in polynomial time. Further insights can be gained if Nash network verification is treated as a decision problem as follows:

NASH NETWORK VERIFICATION (NNV)
INSTANCE: A network formation game $\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I})$ of $n + 1$ players and a joint strategy $t = (t_i)_{i\in I} \in \mathcal{S}$.
QUESTION: Is t a Nash network in $\mathcal{G}$?

In dealing with NNV, we may restrict ourselves to games $\mathcal{G}$ with integer-valued payoff functions, i.e. games with $u_i(s) \in \mathbb{Z}$ for all $i \in I$, $s \in \mathcal{S}$, where $\mathbb{Z}$ denotes the set of integers. The reduction in the proof of Theorem 3.1 still works under this restriction.

First note that NNV belongs to **co-NP**. Namely, given an instance $X = [\mathcal{G}; t]$ of NNV^c (and of NNV), individual strategies $s_j \in S_j,\ j \in I$, can serve as certificates. Given the instance $X = [\mathcal{G}; t]$ and a certificate s_j, it takes quadratic time to calculate the payoffs $u_j(t)$ and $u_j(s_j, t_{-j})$ and to answer the question

$$u_j(s_j, t_{-j}) > u_j(t)\,? \tag{3}$$

If X is a "yes" instance of NNV^c, then the answer will be "yes" for some certificates. If X is a "no" instance of NNV^c, the answer is "no" for all certificates. Hence NNV$^c \in$ **NP** and NNV $\in$ **co-NP**. It follows that

Proposition 4.1 NNV$^c \preceq_p$ GR.

Proof NNV$^c \in$ **NP** and, by Theorem 3.1, GR is **NP**-complete. Therefore, NNV$^c \preceq_p$ GR by the definition of **NP**-completeness. □

By Proposition 4.1, there exists a polynomial-time reduction from NNV^c to GR. The existence of such a reduction implies:

$$\text{If GR} \in \mathbf{P}\text{, then NNV}^c \in \mathbf{P}. \tag{4}$$

An explicit reduction from NNV^c to GR may be found by concatenation of several reductions in the literature, for instance reducing NNV^c to SAT, SAT to SUBSET SUM (possibly via a chain of reductions), and SUBSET SUM to GR. Such a reduction, while polynomial-time, may be rather complicated. Therefore, it proves instructive to provide a second proof of (4) which demonstrates how a deterministic algorithm to solve GR can be used as a sub-routine in the computation of a solution of NNV^c. Furthermore, the argument of the second proof holds for other classes of finite games with polynomial input complexity where payoffs can be computed in polynomial time, regardless of whether GR is **NP**-complete.

To this end, it proves useful to set $B_j = u_j(t) + 1$ and to reformulate question (3) as: Does

$$u_j(s_j, t_{-j}) \geq B_j \tag{5}$$

hold? This reformulation relies on the restriction to integer-valued payoff functions. It offers a specific way of formulating NNV^c. Given an INSTANCE $X = [\mathcal{G}; t]$, the QUESTION is: *Do there exist $j \in I$ and $s_j \in S_j$ such that (5) holds?* The latter is reminiscent of GR. Namely, given an instance $[\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B_j]$ of GR, the question is: *Does there exist $s_j \in S_j$ such that (5) holds?* Exploiting the similarity of the two questions, we are going to provide an alternative, and more direct proof of (4).

ALTERNATIVE PROOF OF (4). Suppose M is a deterministic algorithm that solves GR. One can construct an algorithm M$'$ that solves NNV^c as follows. Given an instance $X = [\mathcal{G}; t]$ of NNV^c with $\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}),\ t = (t_0, \ldots, t_n)$, commence with

ST 0: Calculate $B_0 = u_0(t) + 1$ and apply M to the instance
$X_0 = [\mathcal{G}, 0, t_{-0}, B_0]$ of GR. If M answers "yes", output "yes" and halt.
Else, proceed to ST 1.

ST 1: Calculate $B_1 = u_1(t) + 1$ and apply M to the instance
$X_1 = [\mathcal{G}, 1, t_{-1}, B_1]$ of GR. If M answers "yes", output "yes" and halt.
Else, proceed to ST 2.

....

....

ST n: Calculate $B_n = u_n(t) + 1$ and apply M to the instance
$X_n = [\mathcal{G}, n, t_{-n}, B_n]$ of GR. If M answers "yes", output "yes" and halt.
Else, output "no" and halt.

X is a "yes" instance of NNV^c if and only if at least one of the instances X_j, $j = 0, \ldots, n$ is a "yes" instance of GR. Therefore, M$'$ solves NNV^c.

M$'$ performs at most $n + 1$ of the steps ST 0, ..., ST n. Each step ST j can be performed in polynomial time, if M is a polynomial-time algorithm. Therefore, NNV^c can be solved in polynomial time, if GR can be solved in polynomial time. □

The fact that NNV $\in$ **co−NP** plus the premise **NP** $\neq$ **co−NP** do not necessarily imply that NNV $\notin$ **NP** $\cap$ **co−NP**. To prove that NNV $\notin$ **NP**, we need an intermediate result, Lemma 4.2, which states that NNV $\in$ **NP** would imply BR^{*c} $\in$ **NP**.

The proof of Theorem 3.1 and Corollary 3.2 shows **NP**-completeness of the decision problem BR*, the restriction of BR to instances $[\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ where player j has a quadratic cost function and all other players have linear cost functions.

The proof of Corollary 3.3 provides a polynomial-time reduction from BR* to NN that assigns to every instance $X^0 = [\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ of BR* an instance $X' = f(X^0)$ of NN with the following properties:

(a) $X' = [\mathcal{G}' = (I, (S_i)_{i\in I}, (u'_i)_{i\in I}), B]$.

(b) $s_j \in S_j$ is a best response against t_{-j} in $\mathcal{G}$, with payoff $u_j(s_j, t_{-j}) \geq B$
if and only if
(s_j, t_{-j}) is a Nash network of $\mathcal{G}'$ with $u'_i(s_j, t_{-j}) \geq B$ for all $i \in I$
if and only if
(s_j, t_{-j}) is a Nash network of $\mathcal{G}'$ with $u'_j(s_j, t_{-j}) \geq B$.

(c) If s is a Nash network of $\mathcal{G}'$, then $s_{-j} = t_{-j}$.

These properties yield

Lemma 4.2 *If* NNV $\in$ **NP**, *then* BR^{*c} $\in$ **NP**.

Proof Suppose NNV is in **NP**. Then there exists a deterministic verification algorithm M that generates an output $M(X, z) \in$ {`accept`, `non-accept`} for every instance $X = [\mathcal{G}; t]$ of NNV and every certificate z. Further, if X is a "yes" instance of NNV, then M outputs in polynomial time $M(X, z) =$ `accept` for some z. If X is a "no" instance of NNV, then $M(X, z) =$ `non-accept` for all z.

Now let $X^0 = [\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ be an instance of BR* and BR^{*c}. Let f be the polynomial-time reduction from BR* to NN used in the proof of

Corollary 3.3, so that (a) – (c) hold. Let $X' = f(X^0)$, $X' = [\mathcal{G}'; B]$. According to (b) and (c), X^0 is a "yes" instance of BR^{*c} if and only if there exists $s_j \in S_j$ such that (s_j, t_{-j}) is a Nash equilibrium of $\mathcal{G}'$ and $u'_j(s_j, t_{-j}) < B$.

Next consider certificates for X^0 of the form (s_j, z) with $s_j \in S_j$. Consider the verification algorithm M^0 that generates outputs $M^0(X^0, (s_j, z))$ in the following way:

(i) If $M([\mathcal{G}'; (s_j, t_{-j})], z) = \texttt{non-accept}$,
then $M^0(X^0, (s_j, z)) = \texttt{non-accept}$.
(ii) If $M([\mathcal{G}'; (s_j, t_{-j})], z) = \texttt{accept}$ and $u'_j(s_j, t_{-j}) \geq B$,
then $M^0(X^0, (s_j, z)) = \texttt{non-accept}$.
(iii) If $M([\mathcal{G}'; (s_j, t_{-j})], z) = \texttt{accept}$ and $u'_j(s_j, t_{-j}) < B$,
then $M^0(X^0, (s_j, z)) = \texttt{accept}$.

Complete the definition of M^0 by setting:

(iv) $M^0(X^0, z^0) = \texttt{non-accept}$, if z^0 is a certificate which is not of the form (s_j, z).

In case $X^0 = [\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ is a "yes" instance of BR^{*c}, there exists $s_j \in S_j$ such that (s_j, t_{-j}) is a Nash equilibrium of $\mathcal{G}'$ and $u'_j(s_j, t_{-j}) < B$. Therefore, $X = [\mathcal{G}'; (s_j, t_{-j})]$ is a "yes" instance of NNV. Hence there exists a certificate z such that M outputs in polynomial time $M(X, z) = \texttt{accept}$. Since $u'_j(s_j, t_{-j}) < B$, $M^0(X^0, (s_j, z)) = \texttt{accept}$. Since f is a polynomial-time reduction, $M(X, z) = \texttt{accept}$ is obtained in polynomial time, and $u'_j(s_j, t_{-j})$ can be computed in quadratic time, the output $M^0(X^0, (s_j, z)) = \texttt{accept}$ is obtained in polynomial time.

In case $X^0 = [\mathcal{G} = (I, (S_i)_{i\in I}, (u_i)_{i\in I}), j, t_{-j}, B]$ is a "no" instance of BR^{*c}, then for each $s_j \in S_j$:

- Either (s_j, t_{-j}) is not a Nash equilibrium of $\mathcal{G}'$ and, hence, $M([\mathcal{G}'; (s_j, t_{-j})], z) = \texttt{non-accept}$ and $M^0(X^0, (s_j, z)) = \texttt{non-accept}$ for all z.
- Or (s_j, t_{-j}) is a Nash equilibrium of $\mathcal{G}'$ and $u'_j(s_j, t_{-j}) \geq B$. Hence $M^0(X^0, (s_j, z)) = \texttt{non-accept}$ for all z.

Finally, $M^0(X^0, z^0) = \texttt{non-accept}$ for all certificates z^0 which are not of the form (s_j, z). Therefore, a "no" instance always leads to non-acceptance whereas a "yes" instance leads to acceptance in polynomial time for some certificate. □

Theorem 4.3 *If* **NP** $\neq$ **co-NP***, then* NNV $\notin$ **NP**.

Proof Suppose **NP** $\neq$ **co-NP**. The proof of Theorem 3.1 and Corollary 3.2 shows that BR* is **NP**-complete. Therefore, BR^{*c} $\notin$ **NP**, by Lemma 2.1. Consequently, NNV $\notin$ **NP**, by Lemma 4.2. □

We do not know a polynomial-time reduction from an **NP**-complete problem to NNV. If one could show $P \preceq_p$ NNV for a known **NP**-complete problem P, then Theorem 4.3 would be an immediate consequence of NNV $\in$ **co-NP** and Lemma 2.1.

Because of **P** $\subseteq$ **NP** and **NP** $\neq$ **co-NP** $\Longrightarrow$ **P** $\neq$ **NP**, Theorems 3.1 and 4.3 yield:

Corollary 4.4

(i) *If* $\mathbf{NP} \neq \mathbf{co\text{-}NP}$*, then* NNV $\notin \mathbf{P}$ *and* GR $\notin \mathbf{P}$.
(ii) *If* $\mathbf{P} = \mathbf{NP}$*, then* NNV $\in \mathbf{P}$ *and* GR $\in \mathbf{P}$.

Proof (i) Suppose $\mathbf{NP} \neq \mathbf{co\text{-}NP}$. First, if GR $\in \mathbf{P}$, then $\mathbf{NP} = \mathbf{P}$ by Theorem 3.1—which contradicts $\mathbf{NP} \neq \mathbf{co\text{-}NP}$. Hence GR $\notin \mathbf{P}$. Second, if $\mathbf{NP} \neq \mathbf{co\text{-}NP}$, then NNV $\notin \mathbf{NP}$ by Theorem 2, and therefore, NNV $\notin \mathbf{P}$.

(ii) If $\mathbf{P} = \mathbf{NP}$, then GR $\in \mathbf{P}$ by Theorem 3.1. It follows that there is a deterministic algorithm M that solves GR in polynomial time and, by Proposition 4.1, there exists a deterministic algorithm M$'$ that solves NNV^c in polynomial time. One obtains a deterministic algorithm M$''$ that computes a solution of NNV in polynomial time, by augmenting M$'$ with an simple routine that converts "yes" answers of M$'$ into "no" answers of M$''$ and "no" answers of M$'$ into "yes" answers of M$''$. □

Corollary 4.4 means that there do not exist deterministic algorithms that solve NNV or GR in polynomial time unless $\mathbf{NP} = \mathbf{co\text{-}NP}$. It also means that the computational time complexity of both NNV and GR depends on the answers to the important open questions $\mathbf{NP} \overset{?}{=} \mathbf{co\text{-}NP}$ and $\mathbf{P} \overset{?}{=} \mathbf{NP}$.

If, indeed, the Nash equilibrium property cannot even be verified in reasonable time, the concept itself is somewhat "difficult" to work with, both for the players and the analyst. Then, instead of choosing best responses, players may resort to other behavioral rules. After all, in addition to its importance in computer science, computational complexity provides one of the main motivations for theories of bounded rationality in economics since the pioneering work of Simon (1955). According to Simon (1987), "the term 'bounded rationality' is used to designate choice that takes into account the cognitive limitations of the decision-maker—limitations of both knowledge and computational capacity."

5 Final remarks

That deciding whether there are good (or better) neighbors, good paths, or good Nash networks can be an **NP**-complete or even **NP**-hard problem has been observed in the recent computer science literature; see Fabrikant et al. (2003) for a model where payoffs depend on the shortest paths between players, Anshelevich et al. (2003) where players decide on cost contributions to edges to be formed, Anshelevich et al. (2004) where players choose paths and equally share the cost of each edge they use. What is striking and noteworthy about our results is that **NP**-completeness or **NP**-hardness obtains even though the payoff structure is extremely simple: a player's benefits and costs depend only on the player's direct links.

We observed that BR belongs to **P**, if all cost functions are linear. Similarly, GR, NN and NNV belong to **P**, if all cost functions are linear. In fact, in that case one can find all Nash equilibria (in pure or mixed strategies) in polynomial time. Namely, with linear costs, the game can be decomposed into a collection of $(n^2 - n)/2$ independently played 2×2 bimatrix games. Any Nash equilibrium for the entire game picks one Nash equilibrium from each of the $(n^2 - n)/2$ bimatrix games. Thus an apparently not very drastic difference in the cost functions, linear costs versus quadratic costs (for at least one player), can have a dramatic impact on the computational complexity of decentralized network design. This suggests a potential tradeoff for a social planner (or a society) who has an a priori choice between two technologies (basic infrastructures) for network design, one with high linear costs of link formation and one with low but non-linear costs. If computing time and computing costs are very important, then the technology with linear costs may be preferable even though link formation per se is more costly.

The model of strategic network formation in Baron et al. (2006) is more restrictive than the present one with regard to the benefit functions b_{ij}. It is more general in considering networks consistent with a spatial structure. In Baron et al. (2006), **NP**-hardness of "CONNECTED NN" is shown by means of a reduction from HAMILTONIAN CYCLE for connected graphs. Furthermore, for the associated evolutionary game with asynchronous logit updating, a characterization of the stochastically stable networks is provided.

Existence and properties of Nash networks have been investigated for various variants of Bala and Goyal's (2000) model of strategic network formation; see Bala and Goyal (2000), Haller and Sarangi (2005), Galeotti et al. (2006), Haller et al. (2007). The respective classes of games exhibit "polynomial input complexity" with respect to the number of players. To our knowledge, issues of computational complexity in these games have not been addressed.

The interested reader can check that Eq. (2) in the proof of Theorem 3.1 borrows from Badics and Boros (1998, Theorem 1) who show that the problem of minimization of half-products—a subclass of pseudo-boolean quadratic functions—is **NP**-complete. This suggests that one can easily reduce the problem of minimization of half-products to GR.

References

Anshelevich E, Dasgupta A, Tardos E, Wexler T (2003) Near-optimal network design with selfish agents. In: Proceedings of the 35th annual ACM symposium on the theory of computing (STOC 2003), pp 511–520

Anshelevich E, Dasgupta A, Kleinberg JM, Tardos E, Wexler T, Roughgarden T (2004) The price of stability for network design with fair cost allocation. In: Proceedings of the 45th IEEE symposium on foundations of computer science (FOCS 2004), pp 295–304

Badics T, Boros E (1998) Minimization of half-products. Math Oper Res 23:649–660

Bala V, Goyal S (2000) A non-cooperative model of network formation. Econometrica 68:1181–1229

Baron R, Durieu J, Haller H, Solal P (2006) Complexity and stochastic evolution of dyadic networks. Comput Oper Res 33:312–327

Belleflamme P, Bloch F (2004) Market sharing agreements and collusive networks. Int Econ Rev 45: 387–411

Berninghaus SK, Schwalbe U (1996a) Evolution, interaction, and Nash equilibria. J Econ Behav Org 29: 57–85

Berninghaus SK, Schwalbe U (1996b) Conventions, Local Interaction, and Automata Networks. J Evol Econ 6:297–312

Blume LE (1995) The statistical mechanics of best-response strategy revisions. Games Econ Behav 11: 111–145

Chakrabarti S, Gilles RP (2006) Network potentials, Rev Econ Des 11(1):13–52

Chen X, Deng X (2005a) 3-Nash is PPAD-complete. In: Electronic colloquium on computational complexity, TR05–134

Chen X, Deng X (2005b) Settling the complexity of 2-player Nash-equilibrium. In: Electronic colloquium on computational complexity, TR05–140

Conitzer V, Sandholm T (2003) Complexity results about Nash equilibria. In: Proceedings of the 18th international joint conference on artificial intelligence (IJCAI 2003), 765–771

Daskalakis C, Goldberg PW, Papadimitriou CH (2005) The complexity of computing a Nash equilibrium. In: Electronic colloquium on computational complexity, TR05–115

Daskalakis C, Papadimitriou CH (2005) Three-player games are hard. In: Electronic colloquium on computational complexity, TR05–139

Dutta B, Jackson M (2003) On the formation of networks and groups. In: Dutta B, Jackson M (eds) Networks and groups. Models of strategic formation, Springer, Berlin

Fabrikant A, Luthra A, Maneva E, Papadimitriou CH, Shenker S (2003) On a network creation game. In: Proceedings of the 22nd ACM symposium on principles of distributed computing (PODC 2003), pp 347–351

Galeotti A, Goyal S, Kamphorst J (2006) Network formation with heterogeneous players. Games Econ Behav 54:353–372

Garey MR, Johnson DS (1979) Computers and intractability. A guide to the theory of NP-completeness. Freeman, San Francisco

Gilboa I, Zemel E (1989) Nash and correlated equilibria: some complexity considerations. Games Econ Behav 1:80–93

Haller H, Sarangi S (2005) Nash networks with heterogeneous links. Math Soc Sci 50:181–201

Haller H, Kamphorst J, Sarangi S (2007) (Non-)Existence and scope of Nash networks. Econ Theory 31:597–604

Iacobucci D (ed) (1996) Networks in marketing. Sage Publications, Thousand Oaks

Iacobucci D (1998) interactive marketing and the meganet: networks of networks. J Interact Mark 12(1): 5–16

Iacobucci D, Hopkins N (1992) Modeling dyadic interactions and networks in marketing. J Mark Res 29:5–17

Koller D, Megiddo N (1992) The complexity of two-person zero-sum games in extensive form. Games Econ Behav 4:528–552

Koller D, Megiddo N, Stengel Bvon (1996) Efficient computation of equilibria for extensive two-person games. Games Econ Behav 14:247–259

McKelvey RD, McLennan A (1996) Computation of equilibria in finite games. In: Amman HM, Kendrick DA, Rust J (eds) Handbook of computational economics. Elsevier, Amsterdam

Papadimitriou CH (1994a) Computational complexity. Addison–Wesley Publishing Company, Reading, Mass

Papadimitriou CH (1994b) On the complexity of the parity argument and other inefficient proofs of existence. J Comput Syst Sci 48:498–532

Papadimitriou CH (2005) Computing correlated equilibria in multi-player games. In: Proceedings of the 37th ACM symposium on the theory of computing (STOC 2005), pp 49–56

Papadimitriou CH, Roughgarden T (2005) Computing equilibria in multi-player games. In: Proceedings of the 16th annual ACM-SIAM symposium on discrete algorithms (SODA 2005), pp 82–91

Radner R (1980) Collusive behavior in noncooperative epsilon-equilibria of oligopolies with long but finite lives. J Econ Theory 22:136–154

Radner R (1981) Monitoring cooperative agreements in a repeated principal-agent relationship. Econometrica 49:1127–1148

Sahni S (1974) Computationally related problems. SIAM J Comput 3:262–279

Simon HA (1955) A behavioral model of rational choice. Q J Econ 69:99–118

Simon HA (1987) Bounded rationality. In: Eatwell J, Milgate M, Newman P (eds) The new Palgrave: a dictionary of economics. Macmillan Publishers Limited, Basington; Stockton Press, New York

Topkis DM (1998) Supermodularity and complementarity. Princeton University Press, Princeton

Whitmeyer JM (2002) A deductive approach to friendship networks. J Math Sociol 26:147–165