games

MDPI

Article

An Abstraction-Refinement Methodology for Reasoning about Network Games†

Guy Avni [1,‡], Shibashis Guha [2,‡] and Orna Kupferman [3,*,‡]

1 The Institute of Science and Technology Austria (IST Austria), Am Campus 1, 3400 Klosterneuburg, Austria; guy.avni@ist.ac.at

2 Université Libre de Bruxelles, Avenue Franklin Roosevelt 50, 1050 Bruxelles, Belgium; shibashis.guha@ulb.ac.be

3 School of Computer Science and Engineering, Hebrew University, Jerusalem 91904, Israel

* Correspondence: orna@cs.huji.ac.il

† This paper is an extended version of our paper published in Proceedings of the 26th International Joint Conference on Artificial Intelligence 2017 (IJCAI'17), Melbourne, Australia, 19–25 August 2017.

‡ These authors contributed equally to this work.

Received: 2 May 2018; Accepted: 17 June 2018; Published: 22 June 2018

Abstract: *Network games* (NGs) are played on directed graphs and are extensively used in network design and analysis. Search problems for NGs include finding special strategy profiles such as a Nash equilibrium and a globally-optimal solution. The networks modeled by NGs may be huge. In formal verification, *abstraction* has proven to be an extremely effective technique for reasoning about systems with big and even infinite state spaces. We describe an abstraction-refinement methodology for reasoning about NGs. Our methodology is based on an abstraction function that maps the state space of an NG to a much smaller state space. We search for a global optimum and a Nash equilibrium by reasoning on an under- and an over-approximation defined on top of this smaller state space. When the approximations are too coarse to find such profiles, we *refine* the abstraction function. We extend the abstraction-refinement methodology to labeled networks, where the objectives of the players are regular languages. Our experimental results demonstrate the effectiveness of the methodology.

Keywords: network formation games; abstraction-refinement; Nash equilibrium; social optimum

1. Introduction

Network design is a fundamental and well-studied problem. A game-theoretic approach to the analysis of network design has become especially relevant with the emergence of the Internet, where different users share resources like software or communication channels [1–3]. Network games (NGs, for short) [3–5] constitute a well studied model of non-cooperative games. The game is played among selfish players on a network, which is a directed graph. NGs are used to model resources as edges in a network and the cost involved in sharing these resources. Each player has a source and a target vertex, and a strategy is a choice of a path that connects these two vertices. The cost a player pays is the sum of costs of the edges his path traverses, where a cost of an edge depends on the *load* on it, namely the number of players using the edge. We distinguish between two types of costs. In *cost-sharing* games [3], each edge has a cost and the players that use it split the cost among them, thus increased load decreases the cost. For example, when the costs correspond to prices, users that share a resource also share its price. Then, in *congestion* games [4], increased load increases the cost: each edge has a non-decreasing *cost function* that maps the load on the edge to its cost given this load. For example, when the network models a road system and costs correspond to the travel time, an increased load on an edge corresponds to a traffic jam, increasing the cost of the players that use it.

Since the players attempt to minimize their own costs, they *selfishly* select a path. The focus in game theory is on the *stable* outcomes of a given setting. The most prominent stability concept is that of a *Nash equilibrium* (NE) [6]: a profile (vector of strategies, one for each player) such that no player can decrease his cost by unilaterally deviating from his current strategy; i.e., assuming that the strategies of the other players do not change[1]. By [4], there exists an NE in every NG. A *social optimum* (SO) is a profile that minimizes the sum of the players' costs; thus the one obtained if the players would have obeyed some centralized authority.

Finding SO and NE profiles is a complex and well studied problem that has been a driving force in the development of the algorithmic game theory field. Finding an SO is NP-complete [7,8]. Finding an NE is PLS-complete [9,10] (the complexity class PLS contains local search problems with polynomial time searchable neighborhoods [11], thus it is between P and NP) and the complexity of finding an approximate NE was only recently settled in normal-form games [12,13]. The theoretical tools that have been developed are finding their way to practical applications such as auction design and network routing. For example, recently [14], researchers have suggested to quantify the performance of a traffic network using the *Price of Anarchy* measure [15], which is the ratio between the costs the players pay in the SO and in the most expensive NE. This was done by modeling the network as an NG, and finding the SO and the most expensive NE in it. The high complexity of the problems becomes more acute when we keep in mind that the NGs we study often model huge networks. Indeed, the size of nowadays networks increases in an exponential rate, with networks having an increasing amount of information, and becoming more and more complex [16–18].

The need to cope with very large models is a major research area in *formal verification*. There, we check that a system satisfies its specification by translating the system into some formal model, typically a labeled state-transition graph, and applying model-checking procedures on this model [19]. One of the most successful methodologies for reasoning about the huge state space of systems is *abstraction* [20,21], where we hide some of the information about the system. This enables us to reason about systems that are much smaller, yet it gives rise to approximated solutions. Indeed, hiding of information may result in an *under-approximating* system: one that has fewer behaviors than the original system, or in an *over-approximating* system: one that has more behaviors than the original system. Abstraction methodologies use both types of approximations [22–24].

An important step in methodologies that are based on abstraction is *refinement*. Assume that we find an over-approximation of the system that does not satisfy a universal property. That is, the over-approximation has a forbidden behavior. It may be that this forbidden behavior exists also in the concrete system, in which case the verification algorithm terminates and reports a bug in the system. However, it may also be that the forbidden behavior exists only in the over-approximation, thus the counterexample is *spurious*. Then, one can use the spurious counterexample to refine the over-approximation in a way that eliminates it, and repeat model-checking until either a real counterexample is found, or the over-approximation satisfies the property. This methodology, of *counterexample guided abstraction-refinement* (CEGAR) has proven to be very effective [25].

Abstraction has been studied in the context of extensive-form games. A strategy for a player in an extensive-form game prescribes which action to perform, given the histories of actions played so far. NGs, on the other hand, are "one-shot" games. The questions studied on NGs focus on stable outcome whereas in extensive-form games one often tries to find an optimal strategy for a player. The abstraction studied in [26,27] tries to merge states in the game tree that are indistinguishable for the players. The idea of merging configurations due to hidden information is the key also in abstractions used in formal methods, yet the technical details are very different. Then, in *action abstraction* [27,28], some of the actions of the players are disabled, reducing the state space of reachable configurations in the game tree, which is again not similar to the approach taken here. Finally, formal

[1] Throughout this paper, we consider *pure* strategies, as is the case for the vast literature on NGs.

methods often involve reasoning about multi-player games, and abstraction-refinement methodologies have been studied in this setting [29–32]. Such games model on-going interactions between processes, say a system and its environment, and are infinite-duration games, thus they are again different from the NGs we study here. Moreover, trying to abstract games by standard methods used in formal verification is a known challenge. Indeed, the most conservative form of abstraction is *bisimulation*, which results in a system that is behaviourally equivalent, and it is proven in [33] that NEs are not preserved under bisimulation. Moreover, given the effectiveness of abstractions, researchers have tried to identify games in which NEs are preserved by bisimulation (as is the case, for example, in iterated Boolean games [34]) and have extended the definition of bisimulation to multi-agent systems [29].

In this paper, we introduce and study an abstraction-refinement methodology for reasoning about network games. Given an NG $\mathcal{N}$ defined over a network with a set V of vertices, an *abstraction function* for $\mathcal{N}$ is a function $\alpha : V \to A$, for some set A of abstract vertices. We assume that A is smaller than V, thus the function α induces a partition of V. We define the *under-approximation of $\mathcal{N}$ with respect to α*, denoted $\mathcal{N}^{\downarrow}[\alpha]$, and the *over-approximation of $\mathcal{N}$ with respect to α*, denoted $\mathcal{N}^{\uparrow}[\alpha]$. Both approximations are NGs that have A as their set of vertices. The under- and over-approximation is in the definition of the edges and the cost functions. Intuitively, $\mathcal{N}^{\downarrow}[\alpha]$ is less appealing to the players than $\mathcal{N}$: they have fewer possible profiles, and the profiles that are possible are at least as expensive as the ones that correspond to them in $\mathcal{N}$. Accordingly, the edges under-approximate these in $\mathcal{N}$: there is an edge from an abstract vertex a to an abstract vertex a' if all the concrete vertices that are mapped by α to a have an edge to a concrete vertex that is mapped by α to a' (a.k.a. *must* transitions [21,22]). In addition, the cost of an abstract edge is essentially the maximal cost of a concrete edge that induces it. Dually, $\mathcal{N}^{\uparrow}[\alpha]$ is more appealing to the players: they have more and cheaper profiles than in $\mathcal{N}$. Accordingly, the edges in $\mathcal{N}^{\uparrow}[\alpha]$ over-approximate these in $\mathcal{N}$: there is an edge from a to a' if some concrete vertex that is in a has an edge to some concrete vertex that is mapped to a' (a.k.a. *may* transitions), and the cost of an abstract edge is essentially the minimal cost of a concrete edge that induces it. Traditional abstraction-refinement methodologies in formal verification focus on the transition relation. An extension that takes costs into account has been studied in [35], where the costs of a weighted automaton are also abstracted. Here, we take into account the cost functions as well as the load. Indeed, the merging of edges may lead to a spurious increased load in the abstraction.

We show how $\mathcal{N}^{\downarrow}[\alpha]$ and $\mathcal{N}^{\uparrow}[\alpha]$, which may be significantly smaller than $\mathcal{N}$, can be used in order to reason about the SO and the NE profiles of $\mathcal{N}$. Our methodology is based on the observation that each profile in $\mathcal{N}^{\downarrow}[\alpha]$ can be mapped to at least one profile in $\mathcal{N}$ with a lower or equal cost, and that each profile in $\mathcal{N}$ can be mapped to a profile in $\mathcal{N}^{\uparrow}[\alpha]$ with a lower or equal cost. Hence, for example, the cost of the SO in $\mathcal{N}$ is bounded from above and below by the costs of the SOs in $\mathcal{N}^{\downarrow}[\alpha]$ and $\mathcal{N}^{\uparrow}[\alpha]$, respectively. Moreover, refining α tightens these bounds, so the user can control the trade-off between preciseness and complexity.

A more technically-involved use of the under- and over-approximations is an algorithm we present for finding an NE in $\mathcal{N}$. The algorithm, which can be viewed as the NG-analogue of CEGAR, is based on the notion of an *abstract NE*: we say that a profile P in $\mathcal{N}^{\downarrow}[\alpha]$ is an abstract NE if no player has a beneficial deviation from P even in $\mathcal{N}^{\uparrow}[\alpha]$. Intuitively, an abstract NE has to face two challenges. First, the profile P has to exist in the under-approximation, where fewer strategies exist. In addition, deviations from P are possible in the over-approximation, where more strategies exist, and their cost is lower. Consequently, as we shall formally prove, an abstract NE can direct the search for a concrete NE: once we find an abstract NE P in $\mathcal{N}^{\downarrow}[\alpha]$, it is guaranteed that a concrete NE exists in $\mathcal{N}$ when restricted to profiles that agree with P. Our algorithm finds an abstract NE if one exists and then directs the search for a concrete NE in a much smaller state space. It is, however, not necessary that an abstract NE exists in every abstract game. When a candidate profile in $\mathcal{N}^{\downarrow}[\alpha]$ is an NE in $\mathcal{N}^{\downarrow}[\alpha]$ but is not an abstract NE in $\mathcal{N}$, we use the *spurious deviations* of the players in $\mathcal{N}^{\uparrow}[\alpha]$ in order to refine α, which narrows the search space.

The reachability objectives in NGs enable the players to specify possible sources and targets. Often, it is desirable to refer also to other properties of the selected paths. For example, in a *communication* setting, edges may belong to different providers, and a user may like to specify requirements like "all edges are operated by the same provider" or "no edge operated by AT&T is followed by an edge operated by Verizon". Edges may also have different quality or security levels (e.g., "noisy channel", "high-bandwidth channel", or "encrypted channel"), and again, users may like to specify their preferences with respect to these properties. In *planning* or in *production systems*, nodes of the network correspond to configurations, and edges correspond to the application of actions. The objectives of the players are sequences of actions that fulfill a certain plan, which is often more involved than just reachability [36]; for example "once the arm is up, do not put it down until the block is placed".

In *automata games* (AGs, for short) [37], the edges of the network are labeled by letters from some finite alphabet Σ, and the players have objectives that refer to these labels and are more involved than the reachability objectives of NGs. Specifically, the objective of each player is a language over Σ, specifying a property of the path he should traverse. Note that unlike NGs, a strategy in an AG need not be a simple path. For example, if Σ includes the letters *noise* and *check-sum*, the objective of a player may be to reach some target via a path in which every visit to a noisy edge is followed eventually by a visit to an edge where a check-sum action is performed. Then, a player might be forced to use an edge several times in a strategy. We extended the abstraction-refinement methodology to AGs. We distinguish between two cost-sharing rules that can be applied in AGs. We show that in *uniform AGs*, where the cost of an edge is shared uniformly among all the players that use it, our abstraction-refinement framework remains valid. On the other hand, in *proportional AGs*, where the cost of an edge is shared among the players in proportion to the number of times they have used it, the framework cannot be applied as is. This has to do with the fact that proportional AGs are less stable than NGs, and in fact they need not have an NE [37]. We are still able to use abstract AGs in order to restrict the search for an NE in proportional AGs.

We implemented our methodology and tested its performance on randomly-generated cost-sharing games. We examined the benefit of the abstraction, namely the ratio of the sizes of the concrete game and the abstract NE found in the approximation. We also examined the practicality of our approach, namely the number of CEGAR iterations until an abstract NE is found. We studied these questions for different parameters of the game. Our experimental results demonstrate the efficiency of the methodology. In particular, the results show that the overhead required for abstraction becomes more negligible for larger systems.

The paper is organized as follows. In Section 2, we define NGs, an abstraction-refinement framework for them, and the notion of abstract NEs. In Section 3, we study how the costs of profiles in a concrete NG relate to costs of profiles in its abstraction, and we focus on SO and NE profiles. In Section 4 we describe our methodology for finding an NE in the concrete NG by reasoning about its abstractions, and in Section 5, we describe experimental results that demonstrate the effectiveness of the methodology. Then, in Section 5.1, we extend the setting to AGs, and in Section 6 we discuss the results and point to directions for future research.

2. Preliminaries

2.1. Network Games

A network is a tuple $\langle V, E\rangle$, where V is a set of vertices and $E \subseteq V \times V$ is a set of directed edges. For an integer $k \in \mathbb{N}$, let $[k] = \{1, \ldots, k\}$. A *network game* (NG) is $\mathcal{N} = \langle k, V, E, \{l_e\}_{e\in E}, \langle s_i, t_i\rangle_{i\in[k]}\rangle$, where k is the number of players; $\langle V, E\rangle$ is a network; for $e \in E$, the cost function $l_e : [k] \to \mathbb{R}_{\geq 0}$ maps the load on edge e, namely the number of players that use edge e, to the cost each of them pays for using e with this load; and for $i \in [k]$, the pair $\langle s_i, t_i\rangle \in V \times V$ describes the objective of Player i: traversing $\mathcal{N}$ from the source vertex s_i to the target vertex t_i.

We distinguish between two types of cost functions. In *uniform cost-sharing games* (CS-NGs, for short) the players that use an edge share its cost equally. Formally, each edge e is associated with a weight $w_e \in \mathbb{R}_{\geq 0}$, and for all $x \in [k]$, we have $l_e(x) = \frac{w_e}{x}$. Thus, increasing the load in uniform cost-sharing games decreases the cost of the players. In contrast, in *congestion games* (CON-NGs, for short), the cost functions are non-decreasing, thus increasing the load also increases the cost for each player.

A *strategy* of a player $i \in [k]$ is a simple path π from s_i to t_i. Thus, $\pi = \langle v_1, v_2 \rangle, \langle v_2, v_3 \rangle, \ldots, \langle v_{n-1}, v_n \rangle$, with $v_1 = s_i$, $v_n = t_i$, and $(v_j, v_{j+1}) \in E$ for all $1 \leq j < n$. A *profile* is a tuple of strategies, one for each player. Consider a profile $P = \langle \pi_1, \pi_2, \ldots, \pi_k \rangle$ in the game. We sometimes refer to a path as the set of edges that it traverses, thus $\pi \subseteq E$. For an edge $e \in E$, we use $load_P(e)$ to denote the number of players that traverse the edge e in P. Each player that uses e then pays $l_e(load_P(e))$, and the cost of Player i in P, denoted $cost_i(P)$, is $\sum_{e \in \pi_i} l_e(load_P(e))$. The cost of the profile P, denoted $cost(P)$, is the total cost incurred by all the players, i.e., $cost(P) = \sum_{i=1}^{k} cost_i(P)$. For a profile P and a strategy π of player $i \in [k]$, let $P[i \leftarrow \pi]$ denote the profile obtained from P by replacing the strategy for Player i by π. Given a profile $P = \langle \pi_1, \ldots, \pi_k \rangle$, a *best response* (BR, for short) of Player i in P is a strategy π_i' such that for all strategies π of Player i, we have that $cost_i(P[i \leftarrow \pi_i']) \leq cost_i(P[i \leftarrow \pi])$.

A profile P is a *Nash equilibrium* (NE) of a game $\mathcal{N}$ if no player has an incentive to deviate for $\mathcal{N}$. Formally, a profile P is an NE if for every Player i and every strategy π, we have $cost_i(P[i \leftarrow \pi]) \geq cost_i(P)$. A *social optimum* (SO) of a game $\mathcal{N}$ is a profile that attains the minimal cost over all profiles. We denote by $SO(\mathcal{N})$ the cost of an SO profile; i.e., $SO(\mathcal{N}) = \min_P cost(P)$. An SO can be thought of as an optimal solution imposed by a centralized authority, and need not be an NE.

2.2. Abstraction

Consider an NG $\mathcal{N} = \langle k, V, E, \{l_e\}_{e \in E}, \langle s_i, t_i \rangle_{i \in [k]} \rangle$. We refer to V as the set of *concrete vertices*. Let $T = \{t_1, \ldots, t_k\}$. An *abstraction function* for $\mathcal{N}$ is a function $\alpha : V \to A$, for a set A of *abstract vertices*. We assume that $T \subseteq A$ and that α is such that for all $t_i \in T$, we have $\alpha(t_i) = t_i$ and $\alpha(v) \neq t_i$ for all $v \neq t_i$. We also assume that A is smaller than V, thus the function α induces a partition of V (with a singleton $\{t_i\}$ for each $t_i \in T$). Accordingly, we sometimes refer to abstract vertices as sets of concrete vertices. In particular, when α is clear from the context, we use $v \in a$, for $v \in V$ and $a \in A$, to indicate that $\alpha(v) = a$.

Consider the NG $\mathcal{N}$ and an abstraction function α. We define the *under-* and *over-*approximation of $\mathcal{N}$ formally. Given $\mathcal{N}$, α, and $b \in \{\downarrow, \uparrow\}$, we define $\mathcal{N}^b[\alpha] = \langle k, V, E^b, \{l_e^b\}_{e \in E^b}, \langle \alpha(s_i), \alpha(t_i) \rangle_{i \in [k]} \rangle$, where the under- and over-approximating transition relations $E^{\downarrow}, E^{\uparrow} \subseteq A \times A$, and the under- and over-approximating cost functions $l_e^{\downarrow}$ and $l_e^{\uparrow}$ are defined as follows.

Transition relations: Consider two abstract vertices $a, a' \in A$. Then, $E^{\downarrow}(a, a')$ holds, that is, there is an abstract edge from the abstract vertex a to the abstract vertex a' in the under-approximation $\mathcal{N}^{\downarrow}[\alpha]$, if for every concrete vertex $v \in a$, there is a concrete vertex $v' \in a'$ such that $E(v, v')$. Also, $E^{\uparrow}(a, a')$ holds, that is, there is an abstract edge from the abstract vertex a to the abstract vertex a' in the over-approximation $\mathcal{N}^{\uparrow}[\alpha]$, if there exist concrete vertices $v \in a$ and $v' \in a'$ such that $E(v, v')$. Note that $E^{\downarrow} \subseteq E^{\uparrow}$. For simplicity, we omit self-loops from $E^{\downarrow}$ and $E^{\uparrow}$, as they are not going to be used anyway in strategies. We follow the common terminology in formal verification and refer to the under- and over-approximating edges as *must* and *may* edges, respectively. We extend α to edges in the expected way. Thus, $\alpha(h)$, for an edge $h = \langle v, v' \rangle \in E$, is $\langle \alpha(v), \alpha(v') \rangle$. Note that $\alpha(h)$ is always in $E^{\uparrow}$ and may not be in $E^{\downarrow}$.

Cost Functions: The definition of the under- and over-approximating cost functions depends on the type of $\mathcal{N}$. We first describe the definition and then explain it.

- If $\mathcal{N}$ is a CON-NG, then
  - for every $e \in E^{\downarrow}$ and $x \in [k]$, we have $l_e^{\downarrow}(x) = \max\{l_h(x) : e = \alpha(h)\}$, and
  - for every $e \in E^{\uparrow}$ and $x \in [k]$, we have $l_e^{\uparrow}(x) = \min\{l_h(1) : e = \alpha(h)\}$.

- If $\mathcal{N}$ is a CS-NG, then
  - for every $e \in E^{\downarrow}$ and $x \in [k]$, we have $l_e^{\downarrow}(x) = \max\{l_h(1) : e = \alpha(h)\}$, and
  - for every $e \in E^{\uparrow}$ and $x \in [k]$, we have $l_e^{\uparrow}(x) = \min\{l_h(x) : e = \alpha(h)\}$.

The idea behind the definition is as follows. Recall that in the under-approximation $\mathcal{N}^{\downarrow}[\alpha]$, we want the strategies to be more expensive. This is why we take, in $l_e^{\downarrow}$, the maximal cost of edges that induce e. In CON-NGs, the cost increases with load and hence the cost function $l_e^{\downarrow}$ depends on x since we want more expensive profiles. In CS-NGs, we ignore x and assume that the load is 1. To see why, recall that an abstract edge $e \in E^{\downarrow}$ is obtained by merging several concrete edges. Consequently, the load on e is the sum of the loads on these concrete edges. This load is *fake*: it is only due to the merging of concrete edges and not due to an actual increased load. In CON-NGs, where the cost functions increase with an increased load, fake load goes well with generating more expensive profiles. In CS-NGs, however, increased load decreases the cost, and we have to cancel the fake load. This is done by dividing the load by itself, which bounds the fake load. Recall that in a CS-NG $\mathcal{N}$, each edge $h \in E$ has a weight w_h such that $l_h(x) = \frac{w_h}{x}$. Thus, as $l_h(1) = w_h$, the definition is equivalent to one with $l_e^{\downarrow}(x) = \max\{w_h : e = \alpha(h)\}$.

Dually, the over-approximating cost function aims at providing cheaper strategies. Accordingly, $l_e^{\uparrow}$ depends on the minimum cost function of edges that induce e. Here, we have to cancel fake load in CON-NGs, as fake load increases the cost and may cause the cost of an abstract edge to go beyond the cost of the concrete edges that induce it.

When α is clear from the context, we denote $\mathcal{N}^b[\alpha]$ by $\mathcal{N}^b$. When we refer to the cost of a profile P in $\mathcal{N}^b$, we use the notation $cost^b(P)$, to emphasize that the profile P is in $\mathcal{N}^b$.

The need for having under- and over approximating cost functions has been illustrated in Example 1.

Example 1. *Consider the CON-NG $\mathcal{N}$ appearing in the left of Figure 1. The under- and over-approximating NGs $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$ appear on the right. The abstraction function α maps both v_1 and v_2 to a_1. Thus $E^{\downarrow}$ and $E^{\uparrow}$ coincide in this example. In $\mathcal{N}$, there are two players: Player 1 with the objective (s_1, u_1) and Player 2 with the objective (s_2, u_2). Note that in $\mathcal{N}$, Player 1 has only one strategy while Player 2 has two strategies. Let P be the profile in $\mathcal{N}$ in which Player 2 has the strategy that uses the edge (s_2, v_2). It is easy to see that P is an NE in $\mathcal{N}$.*

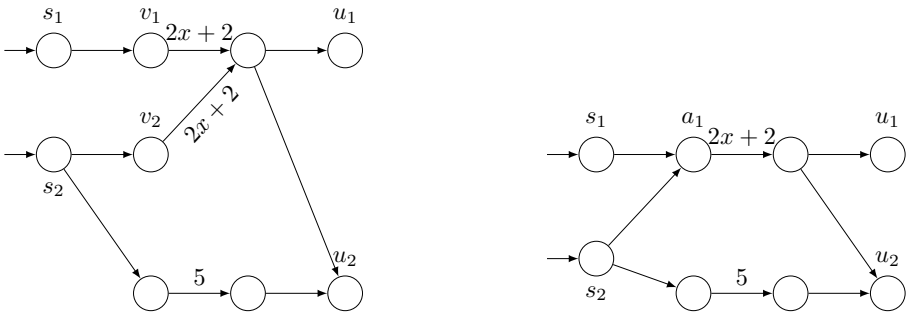

Figure 1. The CON-NG $\mathcal{N}$ (**left**) and its approximations (**right**).

Consider the outgoing edge from a_1, in both $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$. If Player 2 chooses this edge, then it has a load of 2. This is a fake load due to merging of two concrete edges. Consequently, if we take the cost function to be l with no adjustment to the load, we get that the profile $\alpha(P)$, which corresponds to P in $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$ and in which both players go via a_1 is not an NE in $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$. Indeed, by deviating to the strategy that that uses the edge with cost 5, Player 2 reduces his cost to 5. Moreover, the obtained profile is an NE in $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$, which is bad, as it does not correspond to an NE in $\mathcal{N}$.

The function $l^{\uparrow}$ in $\mathcal{N}^{\uparrow}$ cancels the fake load on the abstract edge, and accounts for the fact that single players use the concrete edges that induce it. Using $l^{\uparrow}$, the profile $\alpha(P)$ becomes an NE in $\mathcal{N}^{\uparrow}$, as the cost of using the strategy with the edge (s_2, a_1) is 4.

Below we show a more elaborate example of under- and over-approximations of an NG which will also serve as the running example in this paper.

Example 2. *Consider the concrete NG $\mathcal{N}$ described in the left side of Figure 2. The game is played among three players with a common source vertex s. The target for Player 1 is t. The target for Players 2 and 3 is t'. Consider an abstraction function α that maps the concrete vertices v_2 and v_3 to the abstract vertex a_2, maps v_5 and v_6 to a_5, and does not merge other concrete vertices.*

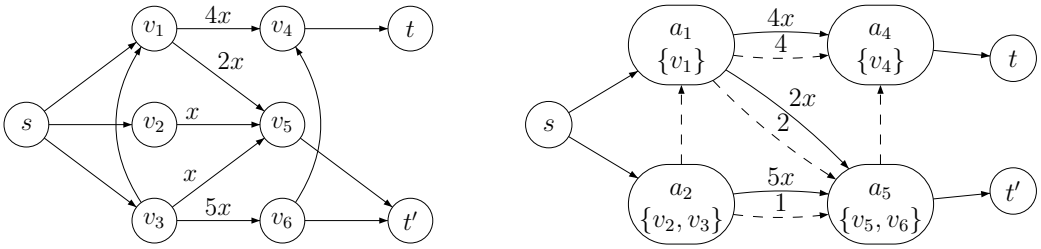

Figure 2. A CON-NG $\mathcal{N}$ (**left**) and its approximations $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$, which share the same state space (**right**). Edges in $E^{\downarrow}$ are solid. Edges in $E^{\uparrow} \setminus E^{\downarrow}$ are dashed. Edges with no specified cost have cost 0.

The under- and over-approximating NGs $\mathcal{N}^{\downarrow}[\alpha]$ and $\mathcal{N}^{\uparrow}[\alpha]$ are described in the right side of the figure. Consider for example the edges from a_2 to a_5. Each of the concrete vertices v_2 and v_3 have an edge to a concrete vertex v_5 or v_6. This is why $(a_2, a_5) \in E^{\downarrow}$. As for the cost, note that there are three concrete edges that induce $\langle a_2, a_5 \rangle$. These are $\langle v_2, v_5 \rangle$, $\langle v_3, v_5 \rangle$ and $\langle v_3, v_6 \rangle$, with cost functions x, x and $5x$, respectively. Accordingly, $l^{\downarrow}_{\langle a_2, a_5 \rangle}(x) = 5x$, which is the maximal cost for one of these three edges, and $l^{\uparrow}_{\langle a_2, a_5 \rangle}(x) = 1$, which is the minimal cost, applied for $x = 1$.

Consider the profile P in $\mathcal{N}^{\downarrow}[\alpha]$ in which Player 1 chooses the path s, a_1, a_4, t and Players 2 and 3 choose the path s, a_2, a_5, t'. In $\mathcal{N}^{\downarrow}[\alpha]$, the cost of Player 1 is $cost^{\downarrow}_1(P) = 4 \times 1 = 4$ and the cost of each of Players 2 and 3 is $cost^{\downarrow}_2(P) = cost^{\downarrow}_3(P) = 5 \times 2 = 10$. The profile P is also a profile in $\mathcal{N}^{\uparrow}[\alpha]$, where $cost^{\uparrow}_1(P) = 4 \times 1 = 4$ and $cost^{\uparrow}_2(P) = cost^{\uparrow}_3(P) = 1$.

Let us emphasize the confusing fact that when we talk about an *under*-approximation, we take the *maximum* cost. This may seem counterintuitive. In order not to get confused, recall that the thing we are approximating is the range and attractiveness of possible profiles. In an under-approximation, we want both fewer and more expensive profiles. Dually, in an *over*-approximation, we take the *minimum* cost, as we want more and cheaper profiles. A similar intuition applies for the adjustment of the load.

2.3. Abstract NE

Recall that in an NE profile in a concrete NG $\mathcal{N}$ is a profile from which no player has an incentive to deviate. In this section we define and discuss stable profiles in the abstraction of $\mathcal{N}$.

Definition 1. [Abstract NE] *A profile $P = \langle \pi_1, \ldots, \pi_k \rangle$ in $\mathcal{N}^{\downarrow}$ is an* abstract NE *if no player has a beneficial deviation from P even in $\mathcal{N}^{\uparrow}$. Formally, for all $i \in [k]$ and strategies $\pi'_i \neq \pi_i$ of Player i in $\mathcal{N}^{\uparrow}$, we have $cost^{\downarrow}_i(P) \leq cost^{\uparrow}_i(P[i \leftarrow \pi'_i])$.*

Intuitively, an abstract NE has to face two challenges. First, the profile P has to exist in the under-approximation, where fewer strategies exist. Second, existence of deviations from P is checked

in the over-approximation, where more strategies may exist, and their cost is lower. Consequently, as we formally prove in Theorem 3, an abstract NE directs the search for a concrete NE: once we find an abstract NE P in $\mathcal{N}^{\downarrow}$, we know that a concrete NE exists in $\mathcal{N}$ when restricted to profiles that agree with P. Formally, given an NG $\mathcal{N}$ and a profile P in $\mathcal{N}^{\downarrow}$, the *restriction of* $\mathcal{N}$ *to* P is the NG $\mathcal{N}_{|P} = \langle k, V_{|P}, E_{|P}, \{l_e\}_{e \in E_{|P}}, \langle s_i, t_i \rangle_{i \in [k]} \rangle$, where $V_{|P} = \{v \in V : \alpha(v)$ appears in a strategy in $P\}$ and $E_{|P} = \{\langle v, v' \rangle \in E : \langle \alpha(v), \alpha(v') \rangle$ appears in a strategy in $P\}$.

Example 3. *Recall the NG and its abstraction that are described in Example 2. Recall the profile P in which Player 1 chooses the path s, a_1, a_4, t and Players 2 and 3 choose the path s, a_2, a_5, t'. Note that P is not an abstract NE. First, Player 1 can deviate to the strategy s, a_1, a_5, a_4, t thereby avoiding the costly edge $\langle a_1, a_4 \rangle$. Note that this strategy is spurious and has no corresponding concrete strategy. Also, Players 2 and 3 can benefit from unilaterally deviating from P to the strategy s, a_1, a_5, t'. Indeed, the cost of Player 2 (and similarly 3) in $\mathcal{N}^{\downarrow}$ is 10 as the load on the must edge $\langle a_2, a_5 \rangle$ is 2, while by deviating to the may edge $\langle a_1, a_5 \rangle$, the cost decreases to 2 as the load is 1. We now show that the abstract game in the right side of Figure 2 does not have an abstract NE.*

Please note that Player 1 has the following strategies in $\mathcal{N}^{\uparrow}$: $A : s, a_1, a_4, t$, $B : s, a_1, a_5, a_4, t$ and $C : s, a_2, a_5, a_4, t$. Please note that the strategies s, a_2, a_1, a_4, t and s, a_2, a_1, a_5, a_4, t of Player 1 are dominated by s, a_1, a_4, t. A is the only strategy of Player 1 in $\mathcal{N}^{\downarrow}$.

Players 2 and 3 have the following strategies in $\mathcal{N}^{\uparrow}$: $E : s, a_1, a_5, t'$ and $F : s, a_2, a_5, t'$,

Please note that the strategies s, a_2, a_1, a_5, t' of Player 2 is dominated by both strategies E and F. Both E and F are valid strategies of Players 2 and 3 in $\mathcal{N}^{\downarrow}$.

Thus we consider the following profiles in $\mathcal{N}^{\downarrow}$ and check if any of them is an abstract NE:

- *$\langle A, E, E \rangle$: Each player has a cost of 4 each. Player 1 can deviate to C and pay 1. Hence this profile is not an abstract NE.*
- *$\langle A, E, F \rangle$: Player 1 pays 4, while Players 2 and 3 have a cost of 2 and 5 respectively. Again Player 1 can deviate to C and pay 1. Hence this profile is not an abstract NE.*
- *$\langle A, F, F \rangle$: Player 1 pays 4, while each of Players 2 and 3 has a cost of 10. Again Player 1 can deviate to C and pay 1. Hence this profile is not an abstract NE.*

Please note that the profile $\langle A, F, E \rangle$, is similar to the profile $\langle A, E, F \rangle$ and hence omitted.

Of special interest, especially in the context of software-defined networks [38], are NGs in which $V = 2^X$ for some set X of variables. Then, abstraction functions are based on predicates on the variables in X. Formally, let $\Psi = \{\psi_1, \ldots, \psi_m\}$ be predicates on X. Thus, each predicate $\psi \in \Psi$ defines a subset $[[\psi]] \subseteq 2^X$ of assignments to X that satisfy it. For example, if $X = \{x_1, \ldots, x_n\}$, then $\psi = x_1 \vee \neg x_2$ is such that $[[\psi]]$ contains all the assignments to X in which $x_1 = \mathit{true}$ or $x_2 = \mathit{false}$. The common practice in *predicate abstraction* is to start with a small set Ψ of predicates, take $A = 2^{\Psi}$, and define $\alpha : 2^X \to 2^{\Psi}$ so that for every $S \in 2^X$, we have $\alpha(S) = \{\psi : S \in [[\psi]]\}$. Thus, an assignment S is mapped to the set of predicates that it satisfies.

The generation of $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$ depends on the way $\mathcal{N}$ is given. When $\mathcal{N}$ is given in a succinct presentation, it is often possible to construct $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$ on top of this succinct presentation. As an example, consider again the case $V = 2^X$, and assume that edge relation E is given by formula θ_E over the set $X \cup X'$ of variables, where $X' = \{x' : x \in X\}$. Thus, for every pair of vertices $v, v' \in V \times V$, we have that $E(v, v')$ if $\theta_E(S_{v,v'})$, where $S_{v,v'}$ is the truth assignment to $X \cup X'$ induced by v and v'. Then, different costs to transitions are defined by different formulas. A simple special case of predicate abstraction is one in which we hide some of the variables. Thus, $A = 2^Y$, for some $Y \subseteq X$, and for all vertices $v \in 2^X$, we have $\alpha(v) = v \cap Y$. Thus, the abstraction hides the variables in $X \setminus Y$. Then, the definition of $E^{\downarrow}$ and $E^{\uparrow}$ is done by a suitable quantification on the hidden variables in θ_E: Let $X \setminus Y = \{z_1, \ldots, z_l\}$. Then, $\theta_E^{\downarrow} = \forall z_1, ..., z_l \exists z'_1, ..., z'_l \theta_E$ and $\theta_E^{\uparrow} = \exists z_1, ..., z_l, z'_1, ..., z'_l \theta_E$.

Example 4. *Consider a huge network of routers, describing external and internal communication among some entities. Each entity may be, for example, an intranet of a company or a large* autonomous system

of routers. Each router is encoded by variables in $X = \{x_1, \ldots, x_n\}$. *The variables* $x_1, \ldots, x_m$, *for some* $m < n$, *maintain the identity of each entity, and the variables* $x_{m+1}, \ldots, x_n$ *maintain the internal identity of each router within its entity. The transitions in the network are described symbolically by some formula* $\theta_E = \theta_{out}(x_1, \ldots, x_m, x'_1, \ldots, x'_m) \vee [\theta_{stay} \wedge \theta_{inter}(x_1, \ldots, x_n, x'_1, \ldots, x'_n)]$, *where* $\theta_{stay} = \wedge_{i=1}^{m}(x_i = x'_i)$. *In other words,* θ_{out} *describes the transitions among the different entities, and* θ_{inter} *describes the internal ones, where* θ_{stay} *guarantees that they are indeed internal. The network depicted in Figure 2 can be described as such a network. The entities are* $v_1, \ldots, v_6$ *and we use three variables to identify them as* $000, \ldots, 101$, *respectively. The transition function* θ_{out} *includes, for example, the disjunct* $(\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \wedge ((\neg x'_1 \leftrightarrow x'_2) \wedge (x'_2 \leftrightarrow x'_3))$ *inducing the transitions from* $v_1 = 000$ *to* $v_4 = 011$ *and* $v_5 = 100$. *Please note that there can be many hidden internal nodes, i.e., we have* $n > 3$. *We now abstract the network further by using the predicates* $\{\neg x_1 \wedge \neg x_2 \wedge \neg x_3, \neg x_1 \wedge (\neg x_2 \leftrightarrow x_3), \neg x_1 \wedge x_2 \wedge x_3, x_1\}$, *which induce respectively the abstract states* a_1, a_2, a_4, *and* a_5 *in the abstract network in Figure 2. For example, only* $v_2 = 001$ *and* $v_3 = 010$ *satisfy the second predicate.*

2.4. Refinement

Consider two abstraction functions $\alpha_1 : V \to A_1$ and $\alpha_2 : V \to A_2$. We say that α_2 *refines* α_1, denoted $\alpha_2 \preceq \alpha_1$, if for all concrete vertices v and v', if $\alpha_2(v) = \alpha_2(v')$, then $\alpha_1(v) = \alpha_1(v')$. That is, the partition of V that is induced by α_2 refines the partition induced by α_1. Please note that whenever α_2 refines α_1, we can view $\mathcal{N}^{\downarrow}[\alpha_1]$ as an under-approximation of $\mathcal{N}^{\downarrow}[\alpha_2]$, and view $\mathcal{N}^{\uparrow}[\alpha_1]$ as an over-approximation of $\mathcal{N}^{\uparrow}[\alpha_2]$. More formally, there is an abstraction function $\alpha : A_2 \to A_1$ such that $\mathcal{N}^{\downarrow}[\alpha_1] = (\mathcal{N}^{\downarrow}[\alpha_2])^{\downarrow}[\alpha]$, and similarly for an over-approximation.

Recall the special interest in NGs in which $V = 2^X$, for some set X of variables, and A is defined by a set Ψ of predicates over X. Then, refinement amounts to enlarging the set Ψ of predicates. In particular, when $A = 2^Y$, for some $Y \subseteq X$, and $\alpha(v) = v \cap Y$, then, for $Y_1 \subseteq Y_2 \subseteq X$, an abstraction $\alpha_2 : V \to 2^{Y_2}$ refines an abstraction $\alpha_1 : V \to 2^{Y_1}$. For example, we can refine the abstraction that is described in Example 4 by replacing the predicate x_1 by two predicates $x_1 \wedge x_3$ and $x_1 \wedge \neg x_3$, which splits the state a_5 in Figure 2 and induces the network in Figure 3.

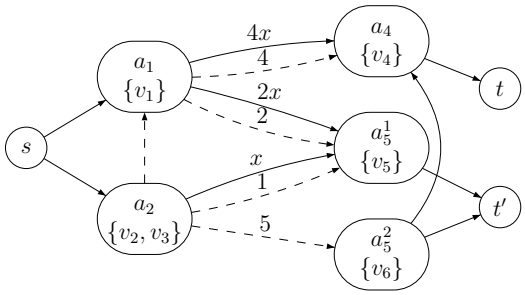

Figure 3. A refinement of the network in Figure 2.

3. On Abstract SOs and NEs

In this section we study the theoretical properties of abstraction in NGs and show how reasoning about the (much smaller) under- and over-approximations of an NG $\mathcal{N}$ can be used for bounding the cost of an SO and for directing the search for an NE in $\mathcal{N}$. We first relate strategies and profiles in $\mathcal{N}$ with strategies and profiles in its approximations.

Consider a network $\mathcal{N}$ and a strategy π of Player i in $\mathcal{N}$. The strategy $\alpha(\pi)$ that corresponds to π in $\mathcal{N}^{\uparrow}$ is obtained from π by replacing each concrete edge h by the abstract edge $\alpha(h)$, and removing cycles in the obtained path in $\mathcal{N}^{\uparrow}$. Please note that by the definition of $E^{\uparrow}$, the edge $\alpha(h)$ exists in $\mathcal{N}^{\uparrow}$. Formally, we define $\alpha(\pi)$ as follows. Let $\pi = \langle v_1, v_2\rangle, \langle v_2, v_3\rangle, \ldots, \langle v_{n-1}, v_n\rangle$. We first define $\alpha'(\pi) = \langle \alpha(v_1), \alpha(v_2)\rangle, \langle \alpha(v_2), \alpha(v_3)\rangle, \ldots, \langle \alpha(v_{n-1}), \alpha(v_n)\rangle$. Then, $\alpha(\pi)$ is obtained from $\alpha'(\pi)$ by removing cycles; that is, by repeatedly removing subsequences $\langle \alpha(v_j), \alpha(v_{j+1})\rangle, \ldots, \langle \alpha(v_{j+m}),$

$\alpha(v_{j+m+1})\rangle$ with $\alpha(v_j) = \alpha(v_{j+m+1})$. A profile $P = \langle \pi_1, \ldots, \pi_k \rangle$ in $\mathcal{N}$ corresponds to the profile $\alpha(P) = \langle \alpha(\pi_1), \ldots, \alpha(\pi_k) \rangle$ in $\mathcal{N}^{\uparrow}$.

Consider now a strategy $\pi = \langle a_1, a_2 \rangle, \langle a_2, a_3 \rangle, \ldots, \langle a_{n-1}, a_n \rangle$ of Player i in $\mathcal{N}^{\downarrow}$. By the definition of $E^{\downarrow}$, for every concrete vertex v with $\alpha(v) = a_1$, and in particular for s_i – the source vertex of Player i, there is a path in $\mathcal{N}$ from v to some vertex v' with $\alpha(v') = a_{n-1}$. Also, by the definition of $\mathcal{N}^{\downarrow}$, we have that $a_n = t_i$ – the target vertex of Player i[2], and for all the concrete vertices v' with $\alpha(v') = a_{n-1}$, we have $E(v', t_i)$. Hence, the strategy π in $\mathcal{N}^{\downarrow}$ induces at least one path $\pi' = \langle v_1, v_2 \rangle, \langle v_2, v_3 \rangle, \ldots, \langle v_{n-1}, v_n \rangle$ in $\mathcal{N}$ such that $v_1 = s_i$ and $v_n = t_i$. Let $\alpha^{-1}(\pi)$ be the nonempty set of these paths. Finally, a profile $P = \langle \pi_1, \ldots, \pi_k \rangle$ in $\mathcal{N}^{\downarrow}$ corresponds to the set $\alpha^{-1}(P)$ of profiles $P' = \langle \pi'_1, \ldots, \pi'_k \rangle$ in $\mathcal{N}$ in which for all $i \in [k]$, we have $\pi'_i \in \alpha^{-1}(\pi_i)$.

We now relate the costs of corresponding profiles in $\mathcal{N}$, $\mathcal{N}^{\downarrow}$, and $\mathcal{N}^{\uparrow}$.

Lemma 1. *Consider an NG $\mathcal{N}$ and an abstraction function α.*

1. *For every profile P in $\mathcal{N}$, the profile $\alpha(P)$ in $\mathcal{N}^{\uparrow}$ is such that $cost^{\uparrow}(\alpha(P)) \leq cost(P)$.*
2. *For every profile P in $\mathcal{N}^{\downarrow}$ and profile $P' \in \alpha^{-1}(P)$ in $\mathcal{N}$, we have that $cost(P') \leq cost^{\downarrow}(P)$.*

Proof. We start with the first claim. Let $P = \langle \pi_1, \ldots, \pi_k \rangle$ in $\mathcal{N}$. Consider a strategy π_i. By the definition of the over-approximating cost function $l^{\uparrow}$, we have that for all edges $h \in \pi_i$ and for every load $x \in [k]$, we have that $l^{\uparrow}_h(x) \leq l_{\alpha(h)}(x)$. Indeed, if $\mathcal{N}$ is a CON-NG, then $l^{\uparrow}_{\alpha(h)}(x)$ is the minimum of a set one of whose elements is $l_h(1)$, which is smaller than $l_h(x)$, and if $\mathcal{N}$ is a CS-NG, then $l^{\uparrow}_{\alpha(h)}(x)$ is the minimum of a set one of whose elements is $l_h(x)$. Hence, for every $i \in [k]$, we have that $cost^{\uparrow}_i(\alpha(P)) \leq cost_i(P)$, implying that $cost^{\uparrow}(\alpha(P)) \leq cost(P)$, and we are done.

We proceed to the second claim. Let $P = \langle \pi_1, \ldots, \pi_k \rangle$, with $\pi_i = \langle a_1, a_2 \rangle, \langle a_2, a_3 \rangle, \ldots, \langle a_{n-1}, a_n \rangle$. Consider a strategy $\pi'_i \in \alpha^{-1}(\pi_i)$. Let $\pi'_i = \langle v_1, v_2 \rangle, \langle v_2, v_3 \rangle, \ldots, \langle v_{n-1}, v_n \rangle$. By the definition of the under-approximating cost function $l^{\downarrow}$, we have that for all $1 \leq j < n_i$ and for every load $x \in [k]$, we have that $l_{\langle v_j, v_{j+1} \rangle}(x) \leq l^{\downarrow}_{\langle a_j, a_{j+1} \rangle}(x)$. Indeed, if $\mathcal{N}$ is a CON-NG, then $l^{\downarrow}_{\langle a_j, a_{j+1} \rangle}(x)$ is the maximum of a set one of whose elements is $l_{\langle v_j, v_{j+1} \rangle}(x)$, and if $\mathcal{N}$ is a CS-NG, then $l^{\downarrow}_{\langle a_j, a_{j+1} \rangle}(x)$ is the maximum of a set one of whose elements is $l_{\langle v_j, v_{j+1} \rangle}(1)$, which is greater than $l_{\langle v_j, v_{j+1} \rangle}(x)$. Hence, for every $i \in [k]$, we have that $cost_i(P') \leq cost^{\downarrow}_i(P)$, implying that $cost(P') \leq cost^{\downarrow}(P)$, and we are done. □

Theorem 1. *For every NG $\mathcal{N}$ and abstraction function α, we have $SO(\mathcal{N}^{\uparrow}[\alpha]) \leq SO(\mathcal{N}) \leq SO(\mathcal{N}^{\downarrow}[\alpha])$.*

Proof. We first show that $SO(\mathcal{N}^{\uparrow}[\alpha]) \leq SO(\mathcal{N})$. Consider a profile $P = \langle \pi_1, \ldots, \pi_k \rangle$ in $\mathcal{N}$. By Lemma 1, we have that $cost^{\uparrow}(\alpha(P)) \leq cost(P)$. In addition, by definition, $SO(\mathcal{N}^{\uparrow}[\alpha]) \leq cost^{\uparrow}(\alpha(P))$. Since this is valid for every profile P in $\mathcal{N}$, and in particular for SO profiles, it follows that $SO(\mathcal{N}^{\uparrow}[\alpha]) \leq SO(\mathcal{N})$.

It is left to show that $SO(\mathcal{N}) \leq SO(\mathcal{N}^{\downarrow}[\alpha])$. Consider a profile P in $\mathcal{N}^{\downarrow}$. By Lemma 1, for every profile $P' \in \alpha^{-1}(P)$, we have that $cost(P') \leq cost^{\downarrow}(P)$. Also, $\alpha^{-1}(P)$ is not empty. Since the profiles $P' \in \alpha^{-1}(P)$ are in $\mathcal{N}$, we also have $SO(\mathcal{N}) \leq cost(P')$. Hence, $SO(\mathcal{N}) \leq cost^{\downarrow}(P)$, and we are done. □

Recall that given two abstraction functions α_1 and α_2, if α_2 refines α_1, then we can view $\mathcal{N}^{\downarrow}[\alpha_1]$ as an under-approximation of $\mathcal{N}^{\downarrow}[\alpha_2]$, and view $\mathcal{N}^{\uparrow}[\alpha_1]$ as an over-approximation of $\mathcal{N}^{\uparrow}[\alpha_2]$. Accordingly, Theorem 1 can be viewed as a special case of Theorem 2 below, with α_2 being the most refined abstraction function (that is, $\alpha_2 : V \to V$, with $\alpha_2(v) = v$), and α_1 being the refinement function α studied there.

---

[2] We note that this is the point where we use the fact that $\alpha(t_i)$ is a singleton, for every $i \in [k]$.

Theorem 2. *Consider an NG $\mathcal{N}$ and two abstraction functions α_1 and α_2. If $\alpha_2 \preceq \alpha_1$, then $SO(\mathcal{N}^{\uparrow}[\alpha_1]) \leq SO(\mathcal{N}^{\uparrow}[\alpha_2])$ and $SO(\mathcal{N}^{\downarrow}[\alpha_2]) \leq SO(\mathcal{N}^{\downarrow}[\alpha_1])$.*

Theorem 1 enables us to approximate the cost of an SO in $\mathcal{N}$ using the costs of the SO in the much smaller $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$. We now turn to study how $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$ can be used in order to direct the search for an NE in $\mathcal{N}$.

Theorem 3. *Consider an NG $\mathcal{N}$, an abstraction function α for it, and an abstract NE P in $\mathcal{N}^{\downarrow}[\alpha]$. There exists a profile in $\alpha^{-1}(P)$ that is a concrete NE in $\mathcal{N}$.*

Proof. Let $P = \langle \pi_1, \ldots, \pi_k \rangle$. Recall that $\mathcal{N}_{|P}$ is the NG obtained from $\mathcal{N}$ by restricting it to profiles in $\alpha^{-1}(P)$. Let $P_{NE} = \langle \sigma_1, \ldots, \sigma_k \rangle$ be a profile in $\alpha^{-1}(P)$ such that for every player $i \in [k]$, deviation to each strategy $\sigma'_i \neq \sigma_i$ where $\sigma'_i \in \alpha^{-1}(P)$, is not beneficial for Player i. We prove that P_{NE} is also an NE in $\mathcal{N}$.

Consider a deviation σ'_i of Player i from his strategy σ_i in P_{NE}. Let $P'_{NE} = P_{NE}[i \leftarrow \sigma'_i]$. We distinguish between two cases.

First, if $P'_{NE} \in \alpha^{-1}(P)$, then, by the definition of P_{NE}, the deviation to σ'_i is not beneficial to Player i.

Otherwise, recall that $P_{NE} \in \alpha^{-1}(P)$. Therefore, by the proof of Lemma 1, for all players $i \in [k]$, we have that $cost_i(P_{NE}) \leq cost_i^{\downarrow}(P)$. In addition, since P is an abstract NE, we have that $cost_i^{\downarrow}(P) \leq cost_i^{\uparrow}(P[i \leftarrow \alpha(\sigma'_i)])$. In addition, $P[i \leftarrow \alpha(\sigma'_i)] = \alpha(P_{NE}[i \leftarrow \sigma'_i])$. Hence, by Lemma 1, we have that $cost_i^{\uparrow}(P[i \leftarrow \alpha(\sigma'_i)]) \leq cost_i(P_{NE}[i \leftarrow \sigma'_i])$. It follows that $cost_i(P_{NE}) \leq cost_i(P_{NE}[i \leftarrow \sigma'_i])$, making the deviation non-beneficial. Hence, P_{NE} is an NE in $\mathcal{N}$, and we are done. □

Please note that profiles in $\alpha^{-1}(P)$ can be searched for in $\mathcal{N}_{|P}$. Thus, as we elaborate in Section 4, an NE in $\mathcal{N}$ can be found by a sequence of best response moves restricted to $\mathcal{N}_{|P}$.

4. An Abstraction-Refinement Procedure for Finding an NE

In this section we describe an abstraction-refinement procedure for finding an NE in an NG . The input to the procedure is a concrete NG $\mathcal{N}$ and an abstraction function α for it. Experience in formal verification suggests that abstraction functions that are supplied by users familiar with the network, are the most successful ones. Alternatively, one can start with a coarse abstraction and refine it as we do in Section 5.

The output of the procedure is a concrete NE in $\mathcal{N}$. Since the state space of $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$ is much smaller than that of $\mathcal{N}$, we would like to perform as many as possible computations on the approximating networks. Our procedure (see Figure 4 for an overview) consists of two parts. The first, in which an abstract NE P_α is found, is done entirely in $\mathcal{N}^{\downarrow}$ and $\mathcal{N}^{\uparrow}$, and it is the procedure we have implemented. The second, in which a concrete NE is found, is done in $\mathcal{N}$, restricted to $\mathcal{N}_{|P_\alpha}$. Thus, as is the case of the CEGAR methodology in formal verification, there is no way to avoid $\mathcal{N}$ entirely, yet we can significantly restrict the part of it in which the search proceeds. Moreover, it is possible to refine α and tighten $\mathcal{N}_{|P_\alpha}$ further. In Section 5, we show that the restriction indeed significantly reduces the size of the network.

There are many ways to refine an abstraction; one can work naively, choose an arbitrary abstract vertex and split it in some way, possibly by adding predicates that appear in the description of the network or the strategies. Even such a naive refinement is guaranteed to eventually lead to a solution. The challenge is to choose the refinements cleverly so that a concrete answer is obtained when the approximating networks are still much smaller than the concrete one. In CEGAR, the refinement is guided by a spurious counterexample. We follow this idea by always refining according to some path in the network that points to a spurious behavior of the approximations. We now describe the methodology in detail.

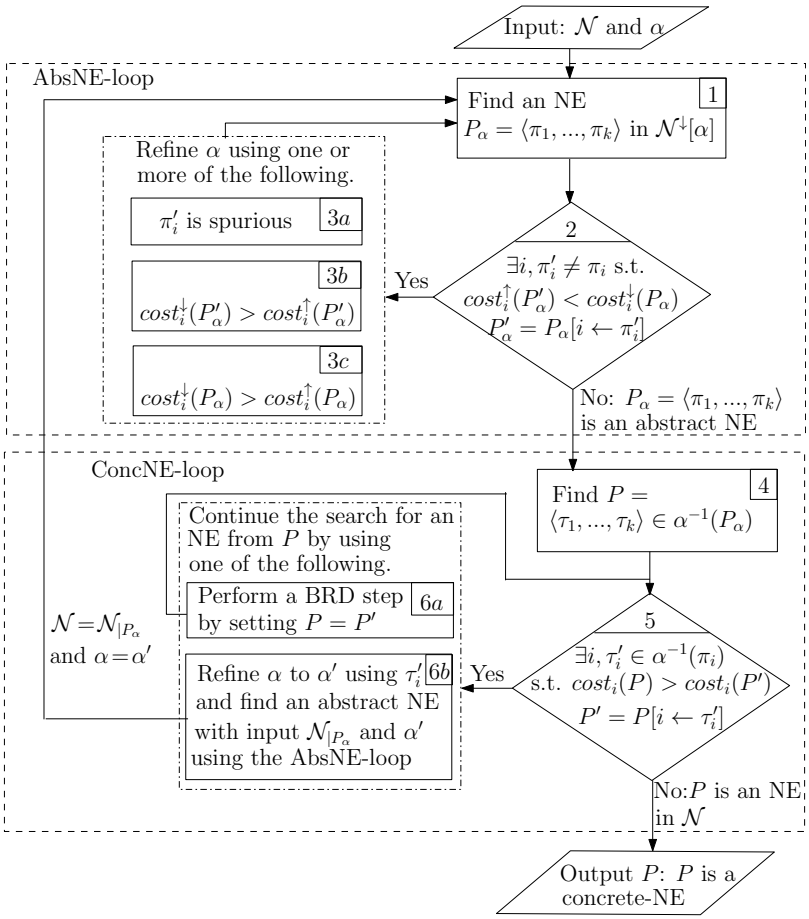

Figure 4. Finding an NE in $\mathcal{N}$.

4.1. Part 1: AbsNE-Loop, Finding an Abstract NE

In the first part, our goal is to find, given $\mathcal{N}$ and α, an abstract NE. Recall that such a profile is an NE in $\mathcal{N}^{\downarrow}$ that is resistant to deviations of the players even in $\mathcal{N}^{\uparrow}$. Since $\mathcal{N}^{\downarrow}$ is an NG, it has an NE. In Step 1 in Figure 4, we find such an NE P_α. This is done by the user's favorite algorithm for finding an NE in NGs. The important point for us is that this is done in the (much smaller) under-approximation of $\mathcal{N}$. Then, in Step 2, we check whether P_α is an abstract NE. Thus, we check whether players have beneficial deviations in $\mathcal{N}^{\uparrow}$. Again, this is done in the (much smaller) over-approximation of $\mathcal{N}$. If no player has a beneficial deviation in $\mathcal{N}^{\uparrow}$, then P_α is an abstract NE, we conclude this part of the procedure, and move to Step 4. Otherwise, there is a player $i \in [k]$ who can benefit from deviating to a strategy π'_i.

We use π'_i in order to guide the refinement. There are several reasons why P_α is an NE in $\mathcal{N}^{\downarrow}$ yet not an abstract NE in $\mathcal{N}$. Step 3 consists of three possible refinement steps among which the user can choose, corresponding to the above different reasons. Let us start with Step 3a. Since π'_i is a path in $\mathcal{N}^{\uparrow}$, there might not be a concrete path in $\mathcal{N}$ that corresponds to it, thus π'_i is a *spurious* path. Consider two adjacent abstract vertices a_1 and a_2 that π'_i traverses. If the edge between a_1 and a_2 is in $E^{\uparrow} \setminus E^{\downarrow}$, we can split a_1 into a'_1 and a''_1 such that a'_1 contains exactly the vertices that have a neighbor in a_2 (similarly we can split a_2). Please note that after refinement, there is a must edge from a'_1 to a_2 and there is not even a may edge between a''_1 to a_2. Since π'_i is spurious, such a candidate vertex is guaranteed to exist (We note that disconnectivity in $\mathcal{N}^{\downarrow}$ can be treated in a similar way.).

We continue to Steps 3b and 3c. They have to do with under- and over-approximations of the cost functions that cause π'_i to be a beneficial deviation in $\mathcal{N}^{\uparrow}$. By the definition of $l^{\downarrow}$ and $l^{\uparrow}$, we know that if $cost_i^{\downarrow}(P_\alpha[i \leftarrow \pi'_i]) > cost_i^{\uparrow}(P_\alpha[i \leftarrow \pi'_i])$, then the path π'_i traverses an abstract edge $e \in E^{\downarrow}$ with load x for which $l_e^{\downarrow}(x) > l_e^{\uparrow}(x)$. Assume that $e = \langle a_1, a_2 \rangle$. In Step 3b, we split a_1 or a_2 in order to tighten this gap. Finally, it may be that the cost of the strategy of Player i in P_α is too big. In Step 3c, we use the strategy π_i that Player i chooses in P_α in order to guide a similar refinement in order to tighten the gap in the costs between $cost_i^{\downarrow}(P_\alpha)$ and $cost_i^{\uparrow}(P_\alpha)$.

A refinement step can be a single refinement or a combination of the refinements that are described above. After completing such a step, we return to Step 1 and find a new NE in the new under-approximating $\mathcal{N}^{\downarrow}$, and repeat the process.

4.2. Part 2: ConcNE-Loop, Finding a Concrete NE

The second part of the procedure gets an abstraction function α as well as an abstract NE $P_\alpha = \langle \pi_1, \ldots, \pi_k \rangle$. The goal is to find a concrete NE P in $\mathcal{N}$ such that $P \in \alpha^{-1}(P_\alpha)$. By Theorem 3, such an NE exists. Recall the best-response dynamics (BRD) algorithm for finding an NE in NGs in which we start with an arbitrary profile, and iteratively allow the players to perform beneficial deviations. We follow this algorithm and start in Step 4 with an arbitrary profile $P = \langle \tau_1, \ldots, \tau_k \rangle$ in $\alpha^{-1}(P_\alpha)$. If P is an NE, we are done. Otherwise, there is a concrete beneficial deviation τ'_i for some Player i. Please note that by Theorem 3, we can restrict the deviations of Player i to paths in $\alpha^{-1}(\pi_i)$. If the size of $\mathcal{N}_{|P_\alpha}$ is small, the user can choose Step 6a and try and find an NE in $\alpha^{-1}(P_\alpha)$ by performing a BRD step. However, when $\mathcal{N}_{|P_\alpha}$ is big, it makes sense to refine the abstraction by choosing Step 6b.

In Step 6b, we let the deviation τ'_i guide the refinement. We look for a vertex v from which τ_i and τ'_i differ, thus from v, one path continues with a vertex v' while the other with v'', where $v' \neq v''$, yet $\alpha(v') = \alpha(v'')$. We refine the abstraction function by splitting $\alpha(v')$ so that v' and v'' are no longer in the same abstract vertex. We would like to have as many must edges as possible between the new vertices. One way to do it is to make v'' a singleton abstract state, but it is also possible to split $\alpha(v')$ as well as $\alpha(v)$ to achieve this. Once we conclude a refinement, we return to the first part of the procedure, and look for an abstract NE with the new abstraction.

Example 5. *Recall the NG and its abstraction that are described in Examples 2 and 3. Consider the profile P_α in which Player 1 chooses the path s, a_1, a_4, t and Player 2 and 3 choose the path $s, a_1, a5, t'$. Please note that while P_α is an NE in $\mathcal{N}^{\downarrow}$, it is not an abstract NE, as Player 2 can deviate to s, a_2, a_5, t'. We refine α according to this deviation, which leads to splitting a_5 into a_5^1 and a_5^2 (see Figure 3). Consider now the profile P_1 in $\mathcal{N}^{\downarrow}$ in which Player 1 chooses the strategy s, a_1, a_4, t and Players 2 and 3 choose the strategy s, a_2, a_5^1, t'. We claim that P_1 is an abstract NE. Indeed, Player 1 has no possible deviation, and Players 2 and 3 each pay 2 for (a_2, a_5^1) and deviating to a path that uses (a_1, a_5^1) or (a_2, a_5^2) does not reduce their costs.*

For the second part of the refinement procedure, we select the concrete profile P in which Player 1 chooses the path s, v_1, v_4, t, and Players 2 and 3 choose the path s, v_2, v_5, t'. This is not an NE in $\mathcal{N}$ as Player 2 can deviate to $\tau'_2 = s, v_3, v_5, t'$. Rather than refining the abstraction, we choose to check if $P[2 \leftarrow \tau'_2]$ is an NE, and indeed it is, so we are done.

Remark 1. *An interesting problem in NGs is to find an NE with "good" social cost [39]. Formally, given an NG $\mathcal{N}$ and a value ν, decide whether there is an NE with cost at most ν. Our procedure can be directed to finding a good NE. Indeed, Lemma 1 shows that the cost of an abstract NE P_α is an upper bound on the cost of every concrete NE in $\alpha^{-1}(P_\alpha)$. Thus, rather than finding an arbitrary abstract NE, we can look for one of a minimal cost, thereby directing the search for a concrete NE to one of a minimal cost too. While this is a harder task than finding an arbitrary abstract NE, it is performed on the much smaller approximation.*

5. Experimental Results

We implemented our methodology and tested the performance of its AbsNE loop on randomly-generated cost-sharing games. We examined the benefit of the abstraction, namely we compared the size of the original game with the game that is truncated to the abstract NE we find. We also examined the practicality of our approach, namely the number of CEGAR iterations until an abstract NE is found. We studied these questions for different parameters of the game; size of the graph, range of weights, and number of players. Our implementations are in Python, we use the library Networkx [40] for graph generation and graph algorithms, and we ran our experiments on a personal computer, Intel Core i5 quad core 1.75 GHz processor, with 8 GB memory. Our results are encouraging: we are able to find an abstract NE relatively easily and it significantly reduces the size of the network making it easier to find a concrete NE.

We generate a random game, given the parameters $n, w, k \in \mathbb{N}$ and $p \in [0,1]$. We use the Erdős-Réyni $G(n,p)$ model to generate the network. Then, for each edge in the graph, we choose at random a cost in a set $\{0,\ldots,w\}$. For each player $i \in [k]$, we choose at random a source vertex s_i and a reachable target vertex t_i. For the initial abstraction, we choose, for $i \in [k]$, a shortest path π_i between s_i and t_i, and we let every vertex that π_i traverses be a singleton abstract state. Thus, in the under approximation, we have at least one path from s_i to t_i. All other concrete vertices are mapped to one abstract state.

We perform three types of experiments. We focus on the number $|V|$ of vertices in the concrete network, the number k of players, and the range $|W|$ of weights on the edges. The number of edges is approximately $1/2 \cdot |V|^2$. We fix two of the parameters and increase the third. In Figure 5, we see how increasing one of the parameters affects the number of iterations of the CEGAR loop. In Figure 6, we compare the sizes of the truncated network, namely, $\alpha^{-1}(P_\alpha)$, and the original one; we show the ratio between the number of vertices in the two networks and the ratio between the number of edges.

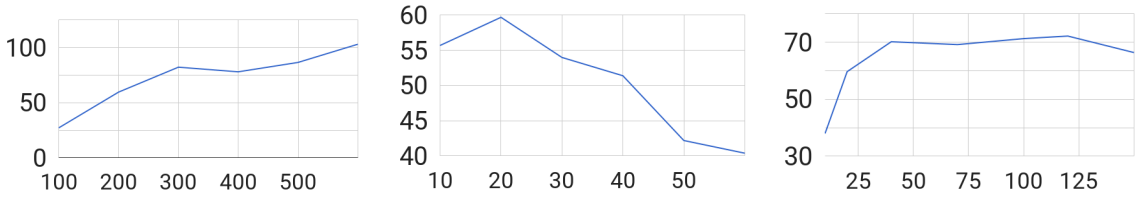

Figure 5. The number of iterations (y-axis) as $|V|$, k, and $|W|$ increase (x-axis).

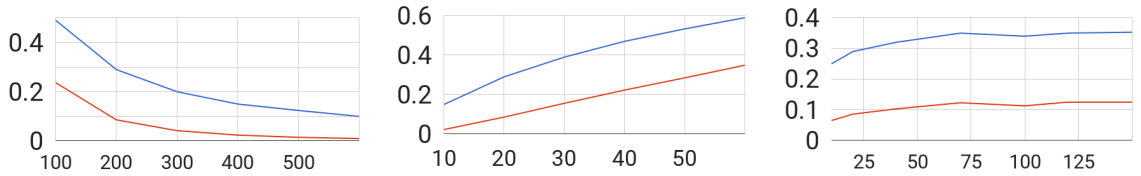

Figure 6. The ratio between the size (vertices in blue, edges in red) of the concrete and truncated networks as $|V|$, k, and $|W|$ increase.

5.1. Analysis of Results

We find the plots with the increasing number of vertices encouraging. As seen in the leftmost plot in Figure 6, since we fix the number of players, the part of the network that is being "used" becomes relatively smaller with increasing $|V|$, and an abstract NE has a good potential to shrink the network. Indeed, the ratios decrease. The downside of this, as seen in Figure 5, is that since the size of the network increases, we need more iterations of the CEGAR loop to find an abstract NE. Next, observe the number of iterations with increasing k, that is, number of players (the middle plots in both figures). Recall how we find an initial abstraction above. When k increases, there is a growing number of concrete vertices that are mapped to singleton abstract states in the initial abstraction.

Thus, the abstraction is closer to the concrete network. On the one hand, as seen in the middle plot in Figure 5, the closer the abstraction is to the concrete network, fewer iterations are needed until an abstract NE is found. On the other hand, as seen in the middle plot in Figure 6, the ratios increase. For increasing $|W|$ (see the right-most plots in the figures), we observe that beyond a particular value of $|W|$, it no longer affects the results.

Automata Games

As discussed in Section 1, the objectives users of a network are often more involved than reachability. In this section we extend the abstraction-refinement methodology to *automata games* (AGs, for short) with cost sharing, which model settings with such richer objectives . AGs are similar to NGs, except that edges of the networks are labeled, and the players have objectives that refer to these labels and are more involved than the reachability objectives of NGs [37].

An AG is played on a *labeled network* $\mathcal{N} = \langle \Sigma, V, \Delta \rangle$, where Σ is a finite alphabet, V is a set of vertices, and $\Delta \subseteq V \times \Sigma \times V$ is a deterministic labeled transition relation, i.e., for $v \in V$ and $\sigma \in \Sigma$, there is at most one $v' \in V$ with $\Delta(v, \sigma, v')$. Consider a path $\pi = e_1, \ldots, e_n$ in $\mathcal{N}$, where, for all $1 \leq i \leq n$, we have $e_i = \langle v_i, \sigma_i, v_{i+1} \rangle \in \Delta$. The word that corresponds to π is $w(\pi) = \sigma_1 \ldots \sigma_n$. An AG is a tuple $\mathcal{A} = \langle k, \Sigma, V, \Delta, \{l_e\}_{e \in \Delta}, \{L_i\}_{i \in [k]} \rangle$, where, for $i \in [k]$, we have that $L_i \subseteq \Sigma^*$ is a regular language that specifies the objective of Player i. Thus, a strategy for Player i is a path π in $\mathcal{N}$ such that $w(\pi) \in L_i$. We assume that the languages L_i are given by means of nondeterministic finite automata over Σ. Our results can be easily extended to other specification formalisms. Please note that unlike NGs, a strategy in an AG need not be simple path. That is, a player might be forced to use an edge several times in a strategy.

We study here cost sharing AGs, and distinguish between two cost-sharing rules that can be applied in AGs. In *uniform AGs*, the cost of an edge is shared uniformly among all the players that use it. Thus, we ignore the fact that different players may use the edge different number of times. This makes the game-theoretical behavior of uniform AGs similar to that of NGs. In *proportional AGs*, introduced and studied in [37], the cost of an edge is shared among the players in proportion to the number of times they have used it. For example, if two players use an edge e with cost c, one uses it once and the second twice, then the first pays $\frac{c}{3}$ for e and the second pays $\frac{2c}{3}$. We define the proportional cost-sharing rule formally in Section 5.3.

5.2. Uniform AGs

Abstracting an AG is similar to abstracting an NG, except that we merge only edges that agree on their label. Formally, consider an AG $\mathcal{A} = \langle k, \Sigma, V, \Delta, \{l_e\}_{e \in \Delta}, \{L_i\}_{i \in [k]} \rangle$ and an abstraction function $\alpha : V \to A$. We construct two abstract AGs $\mathcal{A}^{\downarrow}[\alpha]$ and $\mathcal{A}^{\uparrow}[\alpha]$ as follows. In $\mathcal{A}^{\downarrow}[\alpha]$, we use labeled must edges: for $\sigma \in \Sigma$ and $a, a' \in A$, there is a σ-labeled must edge from a to a' if for every concrete state $c \in a$, there is $c' \in a'$ with $\langle c, \sigma, c' \rangle \in \Delta$. In $\mathcal{A}^{\uparrow}[\alpha]$, we use labeled may edges: for $\sigma \in \Sigma$ and $a, a' \in A$; there is a σ-labeled may edge from a to a' if there exist concrete states $c \in a$ and $c' \in a'$ with $\langle c, \sigma, c' \rangle \in \Delta$. All the other components are as in abstract NGs. For an AG $\mathcal{A}$, and two states $s, t \in V$, we define the (s, t)-language of $\mathcal{A}$, denoted $L^{s,t}(\mathcal{A})$ as the set of words obtained by traversing a path from s to t in $\mathcal{A}$. The use of may and must edges, implies Lemma 2 below (c.f. [41]).

Lemma 2. *Consider an AG $\mathcal{A}$ and an abstraction function $\alpha : V \to A$. For every two states s and t in $\mathcal{A}$, we have that $L^{\alpha(s),\alpha(t)}(\mathcal{A}^{\downarrow}[\alpha]) \subseteq L^{s,t}(\mathcal{A}) \subseteq L^{\alpha(s),\alpha(t)}(\mathcal{A}^{\uparrow}[\alpha])$*

It is not hard to see that all the considerations applied to NGs remain valid for uniform AGs. Indeed, the correspondence between concrete and abstract strategies is preserved, and, by Lemma 2, so does the application of this correspondence in our methodology. Hence, Theorems 1 and 3 are valid also for uniform AGs:

Theorem 4. *Consider a uniform AG $\mathcal{A}$ and an abstraction function α for it.*

- $SO(\mathcal{A}^{\uparrow}[\alpha]) \leq SO(\mathcal{A}) \leq SO(\mathcal{A}^{\downarrow}[\alpha])$.
- *If P is an abstract NE in $\mathcal{A}^{\downarrow}[\alpha]$, then there exists a profile in $\alpha^{-1}(P)$ that is a concrete NE in $\mathcal{A}$.*

5.3. Proportional AGs

We first formally define the costs of the players in a concrete proportional AG. Consider an AG $\mathcal{A} = \langle k, \Sigma, V, \Delta, \{l_e\}_{e\in\Delta}, \{L_i\}_{i\in[k]}\rangle$. Let $P = \langle \pi_1, \pi_2, \ldots, \pi_k\rangle$ be a profile where, for each $i \in [k]$, we have $\pi_i = e_i^1, \ldots, e_i^{\ell_i}$. Let $used_{i,P} : \Delta \to \mathbb{N}$ map each edge to the number of times Player i uses it in profile P. Thus, for $e \in \Delta$, we have $used_{i,P}(e) = |\{j : e_i^j = e\}|$. When $used_{i,P}(e) > 0$, we say that e is in π_i, denoted $e \in \pi_i$. Let $used_P : \Delta \to \mathbb{N}$ denote the total number of times edge e is used in profile P by all players. Thus, for $e \in \Delta$, we have $used_P(e) = \sum_{i\in[k]} used_{i,P}(e)$. The *cost of Player i in profile P* is then $cost_i(P) = \sum_{e\in\pi_i} used_{i,P}(e) \cdot l_e\big(used_P(e)\big)$.

The proportional cost sharing rule cause AGTs to be less stable than NGs:

Theorem 5. *Refs. [37,42] In proportional AGs an NE is not guaranteed to exist, and deciding whether an NE exists in a given game is Σ_2^P-complete.*

Since proportional AGs need not have an NE, the computational question we study is deciding whether a given AG has an NE. While we can abstract proportional AGs as described above, our methodology is targeted for finding an NE and not deciding whether one exists. Indeed, the methodology either outputs an abstract NE, which, recall, is a profile of abstract strategies, or it outputs that no abstract NE exists in the abstraction. We claim that both answers are not helpful for deciding the existence of an NE. While in NGs (and uniform AGs) a concrete NE is guaranteed to exist in the game that is restricted to an abstract NE, in proportional AGs, a concrete NE is not guaranteed to exist at all, let alone in a restriction of the game. Hence, existence of an abstract NE does not imply the existence of a concrete NE. Moreover, existence of an abstract NE that has no concrete NE that is mapped to it, does not imply that a concrete NE does not exist, as such an NE might exist elsewhere. Finally, non-existence of an abstract NE also does not imply the non-existence of a concrete NE.

Below we describe applications where abstraction can still be useful in the context of proportional AGs. First, as SO is independent of the cost sharing rule, the considerations for uniform AGs apply:

Theorem 6. *For every proportional AG $\mathcal{A}$ and an abstraction function α for it, we have that $SO(\mathcal{A}^{\uparrow}[\alpha]) \leq SO(\mathcal{A}) \leq SO(\mathcal{A}^{\downarrow}[\alpha])$.*

Next, we use abstractions in order to restrict the search space when searching for an NE. Recall that the problem of deciding whether an NE exists in proportional AGs is Σ_2^P-complete. This suggests that the optimal algorithm for finding an NE first guesses a profile and then checks whether it is an NE. In order to speed-up the algorithm, we restrict the search space by removing *dominated strategies* using the abstraction.

Intuitively, a strategy π_i for Player i is dominated by a strategy π_i' if no matter how the other players play, it is always better for Player i to use π_i'. Thus, a profile in which Player i uses π_i cannot be an NE and removing dominated strategies does not affect the existence of NE. In other words, a game $\mathcal{A}$ has an NE if the game $\mathcal{A}'$ that is obtained by removing the dominated strategies for all players, has an NE. Formally, for $i \in [k]$, we use π_{-i} to denote a collection of strategies for all players apart from Player i and we denote by $\langle \pi_i, \pi_{-i}\rangle$ the profile in which Player i uses the strategy π_i and the other players choose their strategies in π_{-i}. A strategy π_i for Player i is dominated by a strategy π_i' if for every π_{-i}, we have $cost_i(\langle \pi_i, \pi_{-i}\rangle) > cost_i(\langle \pi_i', \pi_{-i}\rangle)$.

We search for dominated strategies in an AG $\mathcal{A}$ in the abstractions $\mathcal{N}^{\uparrow}[\alpha]$ and $\mathcal{N}^{\downarrow}[\alpha]$. For an abstract strategy $\tau_i = t_1, \ldots, t_n$ for Player i, we need an under- and over-approximations on its

cost. For a concrete edge $e = \langle c, \sigma, c' \rangle$, we use $\alpha(e)$ to denote the abstract may edge $\langle \alpha(c), \sigma, \alpha(c') \rangle$. We extend α also to strategies: for a concrete strategy $\pi = e_1, \ldots, e_n$, we denote the corresponding abstract strategy by $\alpha(\pi) = \alpha(e_1), \ldots, \alpha(e_n)$. For an abstract edge t, we define $w_t^{\downarrow} = \max_{e:\alpha(e)=t} w_e$ and $w_t^{\uparrow} = \min_{e:\alpha(e)=t} w_e$. We define an over-approximation on the cost of τ_i as follows $cost^{\downarrow}(\tau_i) = \sum_{1 \leq j \leq n} w_{t_j}^{\downarrow}$. This is indeed an over-approximation on the cost: Consider a concrete strategy π_i for Player i that is mapped to the abstract may strategy τ_i. Then, for every π_{-i}, we have $cost_i(\langle \pi_i, \pi_{-i} \rangle) \leq cost^{\downarrow}(\tau_i)$.

We now define an under-approximation on the cost of a must strategy τ_i. Let π_{-i} be a collection of $k-1$ concrete strategies and τ_{-i} be the may strategies to which they are mapped. Consider the may profile $P = \langle \tau_i, \tau_{-i} \rangle$. For $j \in [k]$ and a may edge t, the definition of $uses_{j,P}(t)$ and $uses_P(t)$ are similar to the concrete case. Please note that since τ_i is a must path, it is also a may path. We define $cost_i^{\uparrow}(P) = \sum_{t \in \tau_i} w_t^{\uparrow} \cdot uses_{j,P}(t)/uses_P(t)$. We define $cost^{\uparrow}(\tau_i) = \min_{\tau_{-i}} cost_i^{\uparrow}(\langle \tau_i, \tau_{-i} \rangle)$, where the minimum is well-defined since the set of strategies for each player is finite. This is indeed an under-approximation: Consider a concrete strategy π_i for Player i that is mapped to the must strategy τ_i, that is, $\pi_i \in \alpha^{-1}(\tau_i)$. Then, for every π_{-i}, we have $cost_i(\langle \pi_i, \pi_{-i} \rangle) \geq cost_i^{\uparrow}(\langle \tau_i, \tau_{-i} \rangle) \geq cost^{\uparrow}(\tau_i)$.

Definition 2. *Consider an AG $\mathcal{A}$ and an abstraction α for it. We say that a must strategy τ_i for Player i is abstract-dominated by a must strategy τ_i' if $cost^{\downarrow}(\tau_i') < cost^{\uparrow}(\tau_i)$. Let $D_i = \{w(\tau_i) : \tau_i$ is an abstract-dominated strategy$\}$.*

Theorem 7. *Consider a proportional AG $\mathcal{A} = \langle k, \Sigma, V, \Delta, \{l_e\}_{e \in \Delta}, \{L_i\}_{i \in [k]} \rangle$ and an abstraction function α. For $i \in [k]$, let $D_i \subseteq L_i$ be a collection of abstract-dominated strategies for Player i. Then, the game $\mathcal{N}'$ that is obtained by setting the strategies of each Player i to be $L_i \setminus D_i$ has an NE iff $\mathcal{N}$ has an NE.*

Proof. Consider a must strategy τ_i for Player i such that $w(\tau_i) \in D_i$ and let τ_i' be an abstract-dominating strategy. Since both strategies are must paths, there are concrete strategies π_i and π_i' that are mapped to τ_i and τ_i', respectively. We claim that π_i is dominated by π_i', thus a profile in which Player i chooses π_i is not an NE. Let π_{-i} be a choice of strategies for the other players. As in the above, we have $cost_i(\langle \pi_i, \pi_{-i} \rangle) \geq cost^{\uparrow}(\tau_i)$ and $cost_i(\langle \pi_i', \pi_{-i} \rangle) \leq cost^{\downarrow}(\tau_i')$. The claim follows from combining with $cost^{\downarrow}(\tau_i') < cost^{\uparrow}(\tau_i)$. □

6. Discussion and Directions for Future Research

The need to reason about networks of increasing size and complexity calls for the development of new techniques for coping with NGs with large state spaces. Abstraction has proven itself as a very effective method for coping with large state spaces in the context of formal verification. We described an abstraction-refinement methodology for reasoning about NGs. The methodology enables the user to search for two kinds of profiles: Nash equilibria and social optimum. This is done by reasoning about under- and over-approximations of the NG, which are defined over a much smaller state space. When the approximations are too coarse to find such profiles, the user may refine the abstraction. We extended the methodology to labeled networks, where the objectives of the players are regular languages, making it possible to specify properties of the paths along which the reachability objectives are satisfied. We implemented our methodology and our experimental results demonstrate its effectiveness.

This work belongs to a line of works that transfer concepts and ideas between the areas of formal verification, AI, and algorithmic game theory [33]: logics for specifying multi-agent systems [43–45], studies of equilibria in games related to synthesis and repair problems [46–49], and of non-zero-sum games in formal verification [50,51]. Closest to this work are works that apply ideas from formal verification to NGs. This includes an extension of NGs to objectives that are richer than reachability [37], NGs in which the players select their paths dynamically [52], reasoning about real-time in NGs [53,54], and efficient reasoning about NGs that are structured in a hierarchical manner [55]. The latter work is

of special relevance, as it is motivated by the need to cope with large NGs. We believe that further ideas from formal verification should be examined in the context of this challenge. In particular, already in the direction of abstraction and refinement, researchers have studied techniques for finding effective abstraction functions [25] as well as effective refinements for them [24,56]. In the context of NGs, the abstraction function influences not only the structure of the network but also the costs and loads. Consequently, different considerations should be taken when evaluating the effectiveness of an abstraction function. We hope that research on such an evaluation would lead, in addition to techniques for finding effective abstraction functions, also to tighter definitions of the costs in the approximations. Also, especially in the context of AGs, where the networks are labeled, we believe that predicate-based abstractions would serve as a convenient paradigm to study the effectiveness of abstraction functions and their refinements.

Abstraction-refinement is a general concept that is proven to be highly successful in formal verification, where it is used in a search for a counterexample to the correctness of the system [25]. A promising direction would be to apply the basic framework or its quantitative extensions to search problems in AI that need to cope with large inputs, as is already done in planning [57] and in a search for optimal policies in Markov decision processes [58,59]. Extending the abstraction-refinement framework to cope with selfish players, as we do here, is a delicate procedure that depends on the specific game model at hand. Multiagent systems have been studied extensively in AI in a game theoretic framework [60]. Examples of settings that could benefit from an abstraction-refinement framework include prediction of the quality of a given infrastructure (e.g., traffic or internet network) or an auction mechanism [61], finding equilibrium in automated negotiation settings [62], and resource allocation [4], which is a generalization of network games.

Author Contributions: Conceptualization, G.A., S.G. and O.K.; Formal analysis, G.A., S.G. and O.K.; Methodology, G.A., S.G. and O.K.; Software, G.A., S.G. and O.K.; Validation, G.A., S.G. and O.K.; Writing—original draft, G.A., S.G. and O.K.; Writing—review & editing, G.A., S.G. and O.K.

Funding: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013, ERC grant no 278410) and the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE/SHiNE), Z211-N23 (Wittgenstein Award), and M2369-N33 (Meitner fellowship).

Conflicts of Interest: The authors declare no conflict of interest. The funding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

NG	network game
NE	Nash equilibrium
SO	social optimum
CEGAR	counterexample guided abstraction-refinement
CS-NG	cost-sharing network game
CON-NG	congestion network game
AG	automata games

References

1. Fabrikant, A.; Luthra, A.; Maneva, E.; Papadimitriou, C.; Shenker, S. On a Network Creation Game. In Proceedings of the ACM Symposium on Principles of Distributed Computing, Boston, MA, USA, 13–16 July 2003.
2. Albers, S.; Elits, S.; Even-Dar, E.; Mansour, Y.; Roditty, L. On Nash Equilibria for a Network Creation Game. In Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms, Miami, FL, USA, 22–26 January 2006.
3. Anshelevich, E.; Dasgupta, A.; Kleinberg, J.; Tardos, E.; Wexler, T.; Roughgarden, T. The Price of Stability for Network Design with Fair Cost Allocation. *SIAM J. Comput.* **2008**, *38*, 1602–1623. [CrossRef]

4. Rosenthal, R.W. A class of games possessing pure-strategy Nash equilibria. *Int. J. Game Theory* **1973**, *2*, 65–67. [CrossRef]
5. Roughgarden, T.; Tardos, E. How bad is selfish routing? *J. ACM* **2002**, *49*, 236–259. [CrossRef]
6. Nash, J. Equilibrium points in *n*-person games. *Proc. Natl. Acad. Sci. USA* **1950**, *36*, 48–49. [CrossRef] [PubMed]
7. Tardos, E.; Wexler, T. Network Formation Games and the Potential Function Method. In *Algorithmic Game Theory*; Cambridge University Press: Cambridge, UK, 2007; Chapter 19.
8. Meyers, C.A. Network Flow Problems and Congestion Games: Complexity and Approximation Results. Ph.D. Thesis, MIT, Cambridge, MA, USA, 2006.
9. Fabrikant, A.; Papadimitriou, C.; Talwar, K. The complexity of pure Nash equilibria. In Proceedings of the 36th ACM Symp. on Theory of Computing, Chicago, IL, USA, 13–15 June 2004; pp. 604–612.
10. Syrgkanis, V. The Complexity of Equilibria in Cost Sharing Games. In *Lecture Notes in Computer Science, Proceedings of the International Workshop on Internet and Network Economics, Stanford, CA, USA, 13–17 December 2010*; Springer: Berlin, Germany, 2010; Volume 6484, pp. 366–377 .
11. Johnson, D.S.; Papadimtriou, C.H.; Yannakakis, M. How Easy is Local Search? *J. Comput. Syst. Sci.* **1988**, *37*, 79–100. [CrossRef]
12. Babichenko, Y. Query Complexity of Approximate Nash Equilibria. *J. ACM* **2016**, *63*, 36. [CrossRef]
13. Rubinstein, A. Settling the Complexity of Computing Approximate Two-Player Nash Equilibria. In Proceedings of the 57th IEEE Symposium on Foundations of Computer Science, Paris, France, 7–9 October 2016; pp. 258–265.
14. Zhang, J.; Pourazarm, S.; Cassandras, C.G.; Paschalidis, I.C. The Price of Anarchy in Transportation Networks: Data-Driven Evaluation and Reduction Strategies. *Proc. IEEE* **2018**, *106*, 538–553. [CrossRef]
15. Koutsoupias, E.; Papadimitriou, C. Worst-case equilibria. *Comput. Sci. Rev.* **2009**, *3*, 65–69. [CrossRef]
16. Newman, M. The Structure and Function of Complex Networks. *SIAM Rev.* **2003**, *45*, 167–256. [CrossRef]
17. Barabási, A.L. *Linked–How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*; Plume: New York, NY, USA, 2003.
18. Paluch, R.; Lu, X.; Suchecki, K.; Szymański, B.K.; Holyst, J.A. Fast and accurate detection of spread source in large complex networks. *Sci. Rep.* **2018**, *8*, 1–10. [CrossRef] [PubMed]
19. Clarke, E.; Grumberg, O.; Peled, D. *Model Checking*; MIT Press: Cambridge, MA, USA, 1999.
20. Cousot, P.; Cousot, R. Abstract interpretation: A unified lattice model for the static analysis of programs by construction or approximation of fixpoints. In Proceedings of the 4th ACM Symposium on Principles of Programming Languages, Madrid, Spain, 17–23 January 1977; pp. 238–252.
21. Larsen, K. Modal Specifications. In *Lecture Notes in Computer Science, Proceedings of the International Conference on Computer Aided Verification, Grenoble, France, 12–14 June 1989*; Springer: Berlin, Germany, 1989; Volume 407; pp. 232–246.
22. Dams, D.; Gerth, R.; Grumberg, O. Abstract interpretation of reactive systems. *ACM Trans. Programm. Lang. Syst.* **1997**, *19*, 253–291. [CrossRef]
23. Bruns, G.; Godefroid, P. Model Checking Partial State Spaces with 3-Valued Temporal Logics. In Proceedings of the International Conference on Computer Aided Verification, Trento, Italy, 6–10 July 1999; pp. 274–287.
24. Shoham, S.; Grumberg, O. Monotonic Abstraction-Refinement for CTL. In Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Barcelona, Spain, 29 March–2 April 2004; Volume 2988, pp. 546–560.
25. Clarke, E.M.; Grumberg, O.; Jha, S.; Lu, Y.; Veith, H. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM* **2003**, *50*, 752–794. [CrossRef]
26. Gilpin, A.; Sandholm, T. Lossless Abstraction of Imperfect Information Games. *J. ACM* **2007**, *54*, 25. [CrossRef]
27. Brown, N.; Sandholm, T. Simultaneous Abstraction and Equilibrium Finding in Games. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, 25–31 July 2015; pp. 489–496.
28. Gilpin, A.; Sandholm, T.; Sørensen, T.B. A Heads-up No-limit Texas Hold'Em Poker Player: Discretized Betting Models and Automatically Generated Equilibrium-finding Programs. In Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, Estoril, Portugal, 12–16 May 2008; pp. 911–918.

29. Alur, R.; Henzinger, T.; Kupferman, O.; Vardi, M. Alternating refinement relations. In Proceedings of the International Conference on Concurrency Theory, Nice, France, 8–11 September 1998; Volume 1466, pp. 163–178.
30. Henzinger, T.; Majumdar, R.; Mang, F.; Raskin, J.F. Abstract Interpretation of Game Properties. In Proceedings of the International Static Analysis Symposium, Santa Barbara, CA, USA, 29 June–1 July 2000; Volume 1824, pp. 245–252.
31. De Alfaro, L.; Godefroid, P.; Jagadeesan, R. Three-Valued Abstractions of Games: Uncertainty, but with Precision. In Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, Turku, Finland, 17 July 2004; pp. 170–179.
32. Ball, T.; Kupferman, O. An Abstraction-Refinement Framework for Multi-Agent Systems. In Proceedings of the 21st IEEE Symposium on Logic in Computer Science, Seattle, WA, USA, 12–15 August 2006.
33. Gutierrez, J.; Harrenstein, P.; Perelli, G.; Wooldridge, M. Nash Equilibrium and Bisimulation Invariance. In Proceedings of the 28th International Conference on Concurrency Theory, Berlin, Germany, 5–8 September 2017; Volume 85, pp. 1–16.
34. Gutierrez, J.; Harrenstein, P.; Wooldridge, M. Iterated Boolean games. *Inf. Comput.* **2015**, *242*, 53–79. [CrossRef]
35. Avni, G.; Kupferman, O. Making Weighted Containment Feasible: A Heuristic Based on Simulation and Abstraction. In Proceedings of the 23rd International Conference on Concurrency Theory, Newcastle upon Tyne, UK, 4–7 September 2012; Volume 7454, pp. 84–99.
36. Daniele, N.; Guinchiglia, F.; Vardi, M. Improved automata generation for linear temporal logic. In Proceedings of the 11th International Conference on Computer Aided Verification, Trento, Italy, 6–10 July 1999; Volume 1633, pp. 249–260.
37. Avni, G.; Kupferman, O.; Tamir, T. Network-formation games with regular objectives. *Inf. Comput.* **2016**, *251*, 165–178. [CrossRef]
38. Vissicchio, S.; Vanbever, L.; Bonaventure, O. Opportunities and research challenges of hybrid software defined networks. *Comput. Commun. Rev.* **2014**, *44*, 70–75. [CrossRef]
39. Conitzer, V.; Sandholm, T. New complexity results about Nash equilibria. *Games Econ. Behav.* **2008**, *63*, 621–641. [CrossRef]
40. Hagberg, A.A.; Schult, D.A.; Swart, P. Exploring network structure, dynamics, and function using NetworkX. In Proceedings of the 7th Python in Science Conference (SciPy2008), Pasadena, CA, 19–24 August 2008; pp. 11–15.
41. Godefroid, P.; Huth, M.; Jagadeesan, R. Abstraction-based Model Checking using Modal Transition Systems. In Proceedings of the 12th International Conference on Concurrency Theory, Aalborg, Denmark, 20–25 August 2001; Volume 2154, pp. 426–440.
42. Avni, G.; Kupferman, O.; Tamir, T. Congestion and Cost-Sharing Games with Multisets of Resources. In Proceedings of the International Conference on Foundations of Software Technology and Theoretical Computer Science, New Delhi, India, 15–17 December 2015.
43. Alur, R.; Henzinger, T.; Kupferman, O. Alternating-time temporal logic. *J. ACM* **2002**, *49*, 672–713. [CrossRef]
44. Chatterjee, K.; Henzinger, T.A.; Piterman, N. Strategy logic. *Inf. Comput.* **2010**, *208*, 677–693. [CrossRef]
45. Mogavero, F.; Murano, A.; Perelli, G.; Vardi, M.Y. Reasoning About Strategies: On the Model-Checking Problem. *ACM Trans. Comput. Log.* **2014**, *15*, 34. [CrossRef]
46. Chatterjee, K.; Henzinger, T.A.; Jurdzinski, M. Games with secure equilibria. *Theor. Comput. Sci.* **2006**, *365*, 67–82. [CrossRef]
47. Chatterjee, K. Nash Equilibrium for Upward-Closed Objectives. In Proceedings of the 15th Annual Conference on the European Association for Computer Science Logic, Szeged, Hungary, 25–29 September 2006; Volume 4207, pp. 271–286.
48. Fisman, D.; Kupferman, O.; Lustig, Y. Rational Synthesis. In Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Paphos, Cyprus, 20–28 March 2010; Volume 6015, pp. 190–204.
49. Almagor, S.; Avni, G.; Kupferman, O. Repairing Multi-Player Games. In Proceedings of the 26th International 26th Conference on Concurrency, Madrid, Spain, 1–4 September 2015; Volume 42, pp. 325–339.

50. Chatterjee, K.; Majumdar, R.; Jurdzinski, M. On Nash Equilibria in Stochastic Games. In Proceedings of the 13th Annual Conference on the European Association for Computer Science Logic, Karpacz, Poland, 20–24 September 2004; Volume 3210, pp. 26–40.
51. Brihaye, T.; Bruyère, V.; De Pril, J.; Gimbert, H. On Subgame Perfection in Quantitative Reachability Games. *arXiv* **2012**, arXiv:1205.6346.
52. Avni, G.; Henzinger, T.; Kupferman, O. Dynamic Resource Allocation Games. In Proceedings of the International Symposium on Algorithmic Game Theory, Liverpool, UK, 19–21 September 2016; Volume 9928, pp. 153–166.
53. Avni, G.; Guha, S.; Kupferman, O. Timed Network Games. In Proceedings of the 42nd International Symposium on Mathematical Foundations of Computer Science, Aalborg, Denmark, 21–25 August 2017; Volume 83, pp. 1–16.
54. Avni, G.; Guha, S.; Kupferman, O. Timed Network Games with Clocks. In Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science, Liverpool, UK, 27–31 August 2018; Volume 117.
55. Kupferman, O.; Tamir, T. Hierarchical Network Formation Games. In Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Uppsala, Sweden, 24–28 April 2017; Volume 10205, pp. 229–246.
56. Glusman, M.; Kamhi, G.; Mador-Haim, S.; Fraer, R.; Vardi, M. Multiple-Counterexample Guided Iterative Abstraction Refinement: An Industrial Evaluation. In Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Warsaw, Poland, 7–11 April 2003; Volume 2619, pp. 176–191.
57. Pistore, M.; Traverso, P. Planning as Model Checking for Extended Goals in Non-deterministic Domains. In Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, WA, USA, 4–10 August 2001.
58. Filar, J.; Vrieze, K. *Competitive Markov Decision Processes*; Springer: Berlin, Germany, 1996.
59. Sutton, R.S.; Barto, A.G. Reinforcement learning—An introduction. In *Adaptive Computation and Machine Learning*; MIT Press: Cambridge, MA, USA, 1998.
60. Michalak, T.; Rahwan, T.; Wooldridge, M. Strategic Social Network Analysis. In Proceedings of the 31st Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 4841–4845.
61. Sandholm, T. Algorithm for optimal winner determination in combinatorial auctions. *Artif. Intell.* **2002**, *135*, 1–54. [CrossRef]
62. Fatima, S.; Kraus, S.; Wooldridge, M. *Principles of Automated Negotiation*; Cambridge University Press: Cambridge, UK, 2014.