Hierarchical Network Formation Games*

Orna Kupferman † Tami Tamir ‡

January 20, 2017

Abstract

Classical *network-formation games* (NFGs) are played on directed graphs, and are used in network design and analysis. Edges in the network are associated with costs and players have reachability objectives, which they try to fulfill at a minimal cost. When several players use the same edge, they share its cost. The theoretical and practical aspects of NFGs have been extensively studied and are well understood. All studies of NFGs, however, consider an *explicit* representation of the network. In practice, networks are often built in a *hierarchical* manner. Technically, some of the vertices in the network are *boxes*, associated with nested sub-networks, where a sub-network may be "called" by several boxes in the network. This makes hierarchical networks exponentially more succinct than traditional "flat" networks.

We introduce *hierarchical network formation games* (HNFGs) and study theoretical and practical aspects of the hierarchical setting. Different applications call for different cost-sharing mechanisms, which define how edge-formation costs are shared by their users. Indeed, in some applications, cost sharing should refer to the flat expansion of the network and in some it should take into account the hierarchical structure of the network. We study properties of HNFGs like stability and equilibrium inefficiency in the different mechanisms. We also study computational aspects of HNFGs, where the principal question is whether their exponential succinctness with respect to NFGs leads to an exponential increase in the complexity of reasoning about them. This question is analogous to research done in the formal-verification community about the ability to model-check hierarchical systems in their succinct presentation. We show that the picture is diverse and depends on the mechanism applied.

1 Introduction

Network design is a fundamental 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 [21, 1, 19, 8]. In *network-formation games* (NFGs, for short) [8], the network is modeled by a weighted directed graph. The weight of an edge indicates the cost of activating the transition it models, which is independent of the number of

*The research leading to this paper has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 278410, and from The Israel Science Foundation (grant no 1229/10). An extended abstract of this paper appeared in the *Proceedings of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (TACAS), April 2017.

†School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel.

‡School of Computer Science, The Interdisciplinary Center, Herzliya, Israel.

times the edge is used. Players have reachability objectives, each given by a source and a target vertex. A strategy for a player is a path from the source to the target. Under the common fair cost-sharing mechanism, the cost of an edge is shared evenly by the players that use it.

Since the players attempt to minimize their own costs, rather than to optimize some global objective, they *selfishly* select a path instead of being assigned one by a central authority. 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): a profile (vector of strategies, one for each player) such that no player can decrease his cost by unilaterally deviating from his current strategy; that is, assuming that the strategies of the other players do not change.[1] A *best-response* (BR) for a player is a move that results in a profile with a reduced cost for the player. Thus, an NE can be viewed as a profile in which no player has a BR move. A *social optimum* (SO) is a profile that minimizes the total cost of the edges used by all players; thus the one obtained when the players obey some centralized authority.

Research on NFGs involves conceptual questions about them, like the existence of an NE or an analysis of *equilibrium inefficiency*. It is well known that decentralized decision-making may lead to solutions that are sub-optimal from the point of view of society as a whole. The inefficiency incurred due to selfish behavior is quantified according to the *price of stability* (PoS) [8], namely the ratio between the costs of the best NE and the SO, and the *price of anarchy* (PoA) [26, 33], namely the ratio between the costs of the worst NE and the SO. Research also concerns computational problems, like finding an SO, BR moves, and an NE. In NFGs, the picture is well understood. Every NFG has an NE; In a k-player game, the PoS and PoA are $O(\log k)$ and k, respectively; the problem of finding an SO is NP-complete, a single best-response move can be found in polynomial time; and the problem of finding an NE is PLS-complete [24, 37, 32].

To the best of our knowledge, all studies of NFGs consider an *explicit* representation of the network. That is, the network is given by means of its underlying weighted graph, and reasoning about it involves algorithms applied to explicitly-represented graphs. In practice, however, networks are often structured and given in some succinct presentation. This calls for a fresh examination of NFGs. First, the source for the succinctness may require new and more suitable cost-sharing mechanisms. In addition, the computational aspects of NFGs should be examined in terms of their succinct presentation.

In this paper we introduce and study *hierarchical network formation games* (HNFGs). Essentially, HNFGs are NFGs in which some of the vertices in the network may "call" (that is, be substituted by) nested sub-networks. Since a sub-network may be called by several vertices in the network, an HNFG may be exponentially more succinct than the NFG obtained by its "flattening".

Before we describe HNFGs and the challenges they bring with them in more detail, let us survey briefly the analogous research in *model checking*, where the study of succinct presentations and symbolic algorithms is a major research area. In model checking, we verify that a system meets its specification by translating the system to a finite state machine (FSM), translating the specification to a temporal-logic formula, and checking that the FSM satisfies the formula [16]. The translation of a high-level description of a system to an FSM involves a blow-up, and the size of the FSM is typically the computational bottleneck in model-checking algorithms. There are several sources of the blow-up that the translation of systems to FSMs involves. One is the ability of components in the system to work in

[1] Throughout this paper, we consider *pure* strategies, as is the case for the vast literature on cost-sharing games. Unlike mixed strategies, pure strategies may not be random, or drawn from a distribution.

parallel and communicate with each other, possibly using variables [20, 18, 34]. Another source has to do with the ability of a high-level description of a system to reuse the same component in different contexts (say, by calling a procedure). Researchers have studied *hierarchical FSMs*, in which some of the states of the FSM are boxes, which correspond to nested FSMs. The naive approach to model checking such systems is to "flatten" them by repeatedly substituting references to sub-structures by copies of these sub-structures. This, however, results in a flat system that is exponential in the nesting depth of the hierarchical system. In [6], it is shown that for LTL model checking, one can avoid this blow-up altogether, whereas for CTL, one can trade it for an exponential blow-up in the (often much smaller) size of the formula and the maximal number of exits of sub-structures. Likewise, it is shown in [7] that hierarchical parity games can be solved in PSPACE, also leading to a PSPACE model checking algorithm for the μ-calculus. In other words, while hierarchical FSMs are exponentially more succinct than flat FSMs [5], in many cases the complexity of the model-checking problem is not exponentially higher in the hierarchical setting. Thus, there is clear motivation not to flatten the FSM before model checking it. The hierarchical setting is appealing in the context of network design, as many networks are structured in a hierarchical manner. [2] In addition, understanding which types of problems can be solved in the hierarchical setting is of general interest to the formal-verification community.

The fact that box-vertices may be "called" by several vertices in the network calls for new cost-sharing mechanisms – ones that take the hierarchy into account when defining how edge-formation costs are shared by their users. We suggest three different cost-sharing mechanisms. In the *flat* mechanism, the hierarchical structure is flattened and the costs refer to the resulting network. The flat mechanism corresponds to the traditional setting of NFGs, and is suitable in applications in which the traversal of edges corresponds to the utilization of consumable resources. For example, when the network models a hardware design that is built from a library of components, or when the network models a communication system in which local routing is performed by local networks that are composed into a global one. In the *hierarchical* approach, the cost of forming an edge in a sub-network is charged only once, regardless of the number of times it is used in different calls. The hierarchical approach is suitable in applications in which the traversal of edges corresponds to the utilization of non-consumable resources. Thus, repeated calls to a resource do not require its re-formation. For example, when the network models a software design that is built from a library of procedures and functions. The emergence of the OOP programming paradigm makes the hierarchical approach common [29, 27]. In this approach, we study both a *uniform hierarchical* (UH) cost-sharing mechanism, where all players that use an edge share its cost evenly, and a *proportional hierarchical* (PH) cost-sharing mechanism, where the cost of an edge is shared among its users in proportion to their demand. Indeed, each player may use each sub-network a different number of times. In the PH mechanism, this number influences the cost of using the sub-network. Note that the PH mechanism is related to a resource-allocation game in which the strategies of the players are *multisets* of resources [9, 10].

After introducing HNFGs and the possible cost-sharing mechanisms, we study stability and equilibrium inefficiency in the different mechanisms. In particular, we show that while in HNFGs with the flat or UH mechanism, an NE always exists, this is not the case for the PH mechanism. Likewise, while the

[2] We note that different types of hierarchies, mainly ones that refer to a partition of the network to levels, have already been studied. In particular, in [36, 35], it is shown how these levels induce a hierarchical game (also termed "hierarchical NFG", but with the adjective "hierarchical" describing the game rather than the network), leading to a clever decomposition of the game. Our notion of hierarchy is different and refers to nesting of sub-networks. In particular, in earlier work there is no notion of a flat extension, which is the key issue in our games.

PoS and PoA in HNFGs with the flat or UH mechanisms agree with these known for NFGs, HNFGs with the PH mechanism are less stable, and we prove that their PoS may be the number of players. Then, we study the computational aspects of HNFG. The main questions that we answer refer to the ability to reason about an HNFG without first flattening it, which may involve an exponential blow-up. This question is analogous to research done in the formal-verification community about the ability to model-check hierarchical FSMs in their succinct presentation. We observe that the challenge of efficient reasoning about HNFGs starts already with a symbolic presentation of strategies. For the UH and PH mechanisms, we prove that it is sound to restrict attention to *homogeneous* strategies. Intuitively, in such strategies, repeated sub-objectives defined with respect to nested sub-networks are fulfilled in the same way. We show that homogeneous strategies can be represented and operated efficiently. This implies that the problems of finding an SO or a BR move in HNFGs is in NP, and we show matching lower bounds, already for very restricted classes of HNFGs. For the flat mechanism, we focus on HNFGs in which each sub-network has a constant number of exit vertices. We show that for such HNFGs, the problems of finding an SO or an NE are not more complex than these in the non-hierarchical setting.

Many variants of cost-sharing games have been studied. A generalization of the network-formation game of [8] in which players are weighted and a player's share in an edge cost is proportional to its weight is considered in [17], where it is shown that the weighted game does not necessarily have a pure NE. In *congestion games*, sharing of a resource increases its cost. Studied variants of congestion games include settings in which players' payments depend on the resource they choose to use, the set of players using this resource, or both [31, 30, 28, 23]. In some of these variants a pure NE is guaranteed to exist while in others it is not. The three different ideas behind cost sharing, namely flat, UH, and PH, can be combined with other games.

We view this work as another chain in an exciting transfer of concepts and ideas between the areas of game theory and formal verification: logics for specifying multi-agent systems [3, 14], studies of equilibria in games related to synthesis and repair problems [13, 12, 22, 2], an extension of NFGs to objectives that are richer than reachability [9], studies of non-zero-sum games in formal methods [15, 11], augmentation of the problem of synthesis from component libraries with costs [4], and more.

2 Preliminaries

2.1 Hierarchical Graphs

A *weighted graph* is $G = \langle V, E, c\rangle$, where V is a set of *vertices*, $E \subseteq V \times V$ is a set of *directed edges*, and $c : E \to \mathbb{R}_{\geq 0}$ is a *cost function* that maps each edge to a non-negative cost. When $c(e) = 0$, we say that e is *free*. A path in G is a sequence $\rho = e_1, e_2, \ldots, e_m$ of adjacent edges in G. For two vertices $s, t \in V$, we say that ρ is an (s, t)-path if it connects s to t.

A *hierarchical graph* consists of a vector of subgraphs that together compose a graph. A subgraph may be used several times in the composition. Technically, this is done via special vertices, called *boxes*, that are substituted in the composition by other subgraphs. In order to ensure a finite nesting depth of substitutions, the subgraphs are indexed, and a box of a graph can only *call* (that is, be substituted by) subgraphs with a strictly bigger index. Formally, a hierarchical graph is a tuple $\mathcal{G} = \langle G_1, \ldots, G_n\rangle$,

where each subgraph is $G_j = \langle V_j, B_j, in_j, Exit_j, \tau_j, E_j \rangle$, where V_j and B_j are sets of vertices and boxes, respectively. We assume that $B_n = \emptyset$ and that $V_1, \ldots, V_n, B_1, \ldots, B_{n-1}$ are pairwise disjoint. Then, $in_j \in V_j$ is an entry vertex for G_j, and $Exit_j \subseteq V_j$ is a set of exit vertices for G_j. The function $\tau_j : B_j \to \{j+1, \ldots, n\}$ maps each box of G_j to an index greater than j. If $\tau_j(b) = \ell$, we say that the box b is substituted by G_ℓ in G_j. Finally, E_j is an edge relation. Each edge in E_j is a pair $\langle u, v \rangle$ with source u and target v. The source u is either a vertex of G_j, or a pair (b, x), where $b \in B_j$ and $x \in Exit_{\tau_j(b)}$. That is, u may be a box b coupled with an exit vertex of the subgraph by which b is about to be substituted. The target v is a vertex or a box of G_j. Formally, $E_j \subseteq (V_j \cup (\bigcup_{b \in B_j} (\{b\} \times Exit_{\tau_j(b)}))) \times (V_j \cup B_j)$. The *depth* of $\mathcal{G}$ is the number n of subgraphs. A *weighted hierarchical graph* is a hierarchical graph with cost functions $c_j : E_j \to \mathbb{R}_{\geq 0}$ that map the edges in each subgraph to costs.

A subgraph without boxes is *flat*. Every hierarchical graph can be transformed to an equivalent flat graph, referred to as its *flat expansion*, by recursively substituting each box by a copy of the corresponding subgraph. Formally, given a hierarchical graph $\mathcal{G}$, we inductively define for each subgraph G_j its flat expansion $G_j^f = \langle V_j^f, in_j, Exit_j, E_j^f \rangle$, where $V_j^f = V_j \cup (\bigcup_{b \in B_j} (\{b\} \times V^f_{\tau_j(b)}))$. Note that different boxes in G_j can be substituted by the same subgraph. This is why we preserve b as an identifier when we substitute it by the flat expansion of $\tau_j(b)$. The edge relation E_j^f includes the following edges, which we partition into four classes.

- **[Top]:** $\langle u, v \rangle$ such that $u, v \in V_j$ and $\langle u, v \rangle \in E_j$,
- **[Call]:** $\langle u, (b, v) \rangle$ such that $u \in V_j$, $v = in_{\tau_j(b)}$, and $\langle u, b \rangle \in E_j$,
- **[Return]:** $\langle (b, u), v \rangle$ such that $u \in Exit_{\tau_j(b)}$, $v \in V_j$, and $\langle (b, u), v \rangle \in E_j$, and
- **[Internal]:** $\langle (b, u), (b, v) \rangle$ such that $u, v \in V^f_{\tau_j(b)}$ and $\langle u, v \rangle \in E^f_{\tau_j(b)}$.

Note that each edge in E_j^f originates from an edge $\langle u, v \rangle \in E_{j'}$ for some $j' \geq j$. Indeed, in top, call, and return edges, we have that $j' = j$, and in internal edges, we have that j' is the subgraph from which the edge $\langle u, v \rangle$ originates (recursively) in $E^f_{\tau_j(b)}$. Formally, let $E = \bigcup_{1 \leq j \leq n} E_j$ and $E^f = \bigcup_{1 \leq j \leq n} E_j^f$. Then, the function $orig : E^f \to E$ is defined recursively as follows. For a top edge $e = \langle u, v \rangle$ or a return edge $e = \langle (b, u), v \rangle$, we have $orig(e) = e$. For a call edge $e = \langle u, (b, v) \rangle$, we have $orig(e) = \langle u, b \rangle$. Then, for an internal edge $e = \langle (b, u), (b, v) \rangle$, we have $orig(e) = orig(\langle u, v \rangle)$. The graph G_1^f is the flat expansion of $\mathcal{G}$, and we denote it by $\mathcal{G}^f$. For an edge e in $\mathcal{G}^f$, we refer to $orig(e)$ as the *origin of* e in $\mathcal{G}^f$. Consider a path $\rho = e_1, e_2, \ldots, e_m$ in $\mathcal{G}^f$. For a set $\pi \subseteq E$ of edges in $\mathcal{G}$, we say that ρ is *covered by* π if for all $1 \leq i \leq m$, we have $orig(e_i) \in \pi$.

A *multiset* over a set E of elements is a generalization of a subset of E in which each element may appear more than once. For a multiset π over E and an element $e \in E$, we use $\pi(e)$ to denote the number of times e appears in π. For two multisets π_1 and π_2, the union of π_1 and π_2 is the multiset $\pi_1 \cup \pi_2$ in which for all $e \in E$, we have $(\pi_1 \cup \pi_2)(e) = \pi_1(e) + \pi_2(e)$. Then, the difference between p_1 and p_2 is the multiset $\pi_1 \setminus \pi_2$ in which for all $e \in E$, we have $(\pi_1 \setminus \pi_2)(e) = \max\{0, \pi_1(e) - \pi_2(e)\}$. A multiset π is given as a set of its members, with each member e followed by a binary (or decimal) encoding of $\pi(e)$. Accordingly, we define the length of π by $\sum_{e \in \pi} \log \pi(e)$. Consider a path $\rho = e_1, e_2, \ldots, e_m$ in $\mathcal{G}^f$ and a multiset π over E; that is, π is a multiset of edges in $\mathcal{G}$. We say that ρ is *covered by* π if for every edge $e \in E$, the number of edges in ρ whose origin is e is at most the number

of times that e appears in π. Formally, for every $e \in E$, we have that $|\{1 \leq i \leq m : orig(e_i) = e\}| \leq \pi(e)$.

Example 2.1 *Figure 1 presents a weighted hierarchical graph $\mathcal{G} = \langle G_1, G_2 \rangle$ with $\tau_1(b_1) = \tau_1(b_2) = G_2$. The flat expansion $\mathcal{G}^f$ of $\mathcal{G}$ appears on the right.*

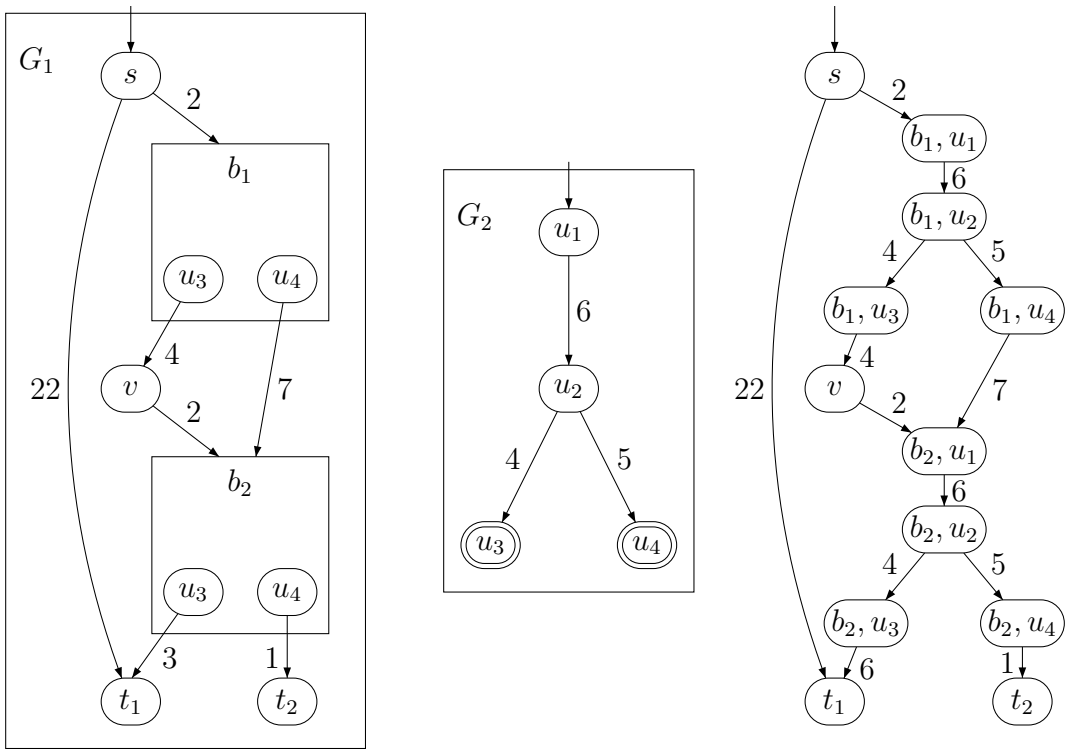

Figure 1: An example of a hierarchical graph and its flat expansion.

The path $\rho = \langle s, (b_1, u_1) \rangle, \langle (b_1, u_1), (b_1, u_2) \rangle, \langle (b_1, u_2), (b_1, u_4) \rangle, \langle (b_1, u_4), (b_2, u_1) \rangle, \langle (b_2, u_1), (b_2, u_2) \rangle, \langle (b_2, u_2), (b_2, u_4) \rangle, \langle (b_2, u_4), t_2 \rangle$ in $\mathcal{G}^f$ is covered by the set $\pi = \{\langle s, b_1 \rangle, \langle (b_1, u_4), b_2 \rangle, \langle (b_2, u_4), t_2 \rangle, \langle u_1, u_2 \rangle, \langle u_2, u_4 \rangle\}$. Note that each of the edges $\langle s, b_1 \rangle, \langle (b_1, u_4), b_2 \rangle$, and $\langle (b_2, u_4), t_2 \rangle$ in π serve as the origin of a single edge in ρ, whereas each of the edges $\langle u_1, u_2 \rangle$ and $\langle u_2, u_4 \rangle$ serve as the origin of two edges in ρ. Accordingly, ρ is covered by the multiset $\pi = \{\langle s, b_1 \rangle^1, \langle (b_1, u_4), b_2 \rangle^1, \langle (b_2, u_4), t_2 \rangle^1, \langle u_1, u_2 \rangle^2, \langle u_2, u_4 \rangle^2\}$.

We define the size of a hierarchical graph $\mathcal{G}$ by $|\mathcal{G}| = \sum_{j=1}^{n} (|V_j| + |B_j|)$. The size of its flat expansion, denoted $|\mathcal{G}^f|$, is the number of vertices in $|\mathcal{G}^f|$. Note that $|\mathcal{G}^f| = \sum_{j=1}^{n} (|V_j| + \sum_{b \in B_j} |G^f_{\tau_j(b)}|)$. It is not hard to see that the hierarchical setting is exponentially more succinct. Formally, we have the following.

Proposition 2.1 *Flattening a hierarchical graph may involve an exponential blow up. That is, $\mathcal{G}^f$ may be exponentially larger than $\mathcal{G}$. In fact, the exponential blow-up applies already to the diameter of the graph, and applies even when all the subgraphs in $\mathcal{G}$ have a single exit vertex.*

Proof: Consider a hierarchical graph $\mathcal{G}$ in which G_n has 2 vertices and every subgraph G_j, for

$j \in [n-1]$, has an entry vertex branching into two boxes, both calling G_{i+1}, and then reaching an exit vertex. We have that $\mathcal{G}$ is of size $4n-2$ whereas $\mathcal{G}^f$ is of size $O(2^n)$. Moreover, if the calls to the two boxes are sequential, the blow-up is in the diameter of $\mathcal{G}^f$. ∎

2.2 Network Formation Games

For an integer $k \in \mathbb{N}$, let $[k] = \{1, \ldots, k\}$. A *network-formation game* (NFG, for short) [8] is $\mathcal{N} = \langle k, G, \langle s_i, t_i \rangle_{i \in [k]} \rangle$, where k is the number of players, $G = \langle V, E, c \rangle$ is a weighted graph, and for each $i \in [k]$, the pair $\langle s_i, t_i \rangle \in V \times V$ describes the objective of Player i, namely forming a path from his source vertex s_i to his target vertex t_i.

A *strategy* of a player $i \in [k]$ is a path from s_i to t_i. A *profile* in $\mathcal{N}$ is a tuple $P = \langle \pi_1, \ldots, \pi_k \rangle$ of strategies for the players. That is, for $1 \leq i \leq k$, we have that π_i is a path from s_i to t_i. Consider a profile $P = \langle \pi_1, \ldots, \pi_k \rangle$. Recall that c maps each edge to a cost, intuitively standing for the cost of its formation. The players aim at fulfilling their objective with minimal cost. Since all costs are positive, we can restrict attention to strategies in which the paths chosen by the players are simple. Then, we can also ignore the order between the edges in the paths and assume that for all $i \in [k]$, we have that $\pi_i \subseteq E$ is a set of edges that compose a path from s_i to t_i. For an edge $e \in E$, we denote by $load_P(e)$, the number of players that use the edge e in P. Formally, $load_P(e) = |\{i : e \in \pi_i\}|$. Players that share an edge also share its formation cost. Thus, the cost of Player i in the profile P is $cost_i(P) = \sum_{e \in \pi_i} \frac{c(e)}{load_P(e)}$. Finally, the cost of a profile P is the sum of the costs of all the players in P. Thus, $cost(P) = \sum_{i \in [k]} cost_i(P)$. Note that $cost(P)$ is equal to the sum of the costs of edges that participate in some strategy in P.

For a profile P and a strategy π of player $i \in [k]$, let $[P_{-i}, \pi]$ denote the profile obtained from P by replacing the strategy for Player i by π. For two strategies π_i and π'_i of Player i, we say that π_i is *dominated by* π'_i, if for every profile P in which Player i uses π_i, we have that $cost_i([P_{-i}, \pi'_i]) \leq cost_i(P)$. A *best response* (BR) for Player i is a strategy π_i that minimizes $cost_i([P_{-i}, \pi_i])$. A profile P is said to be a *(pure) Nash equilibrium* (NE) if none of the players in $[k]$ can benefit from an unilateral deviation from his strategy in P to another strategy. In other words, for every player i and every strategy π that Player i can deviate to from his current strategy in P, it holds that $cost_i([P_{-i}, \pi]) \geq cost_i(P)$. The set of NEs of the game $\mathcal{N}$ is denoted by $\Gamma(\mathcal{N})$.

A *social optimum* (SO) of a game $\mathcal{N}$ is a profile that attains the lowest cost. We denote by $OPT(\mathcal{N})$ the cost of an SO profile; i.e., $OPT(\mathcal{N}) = \min_P cost(P)$. A social optimum may be achieved by a centralized authority and need not be a NE. The following parameters measure the inefficiency caused as a result of the selfish interests of the players. First, the *price of stability* (PoS) [8] of an NFG $\mathcal{N}$ is the ratio between the minimal cost of an NE and the cost of a social optimum of $\mathcal{N}$. That is, $\mathrm{PoS}(\mathcal{N}) = \min_{P \in \Gamma(\mathcal{N})} cost(P)/OPT(\mathcal{N})$. Then, the *price of anarchy* (PoA) [33] of $\mathcal{N}$ is the ratio between the maximal cost of an NE and the cost of the social optimum of $\mathcal{N}$. That is, $\mathrm{PoA}(\mathcal{N}) = \max_{P \in \Gamma(\mathcal{N})} cost(P)/OPT(\mathcal{N})$.

2.3 Hierarchical Network Formation Games

An *hierarchical network-formation game* (HNFG, for short) $\mathcal{N} = \langle k, \mathcal{G}, \langle s_i, t_i \rangle_{i \in [k]} \rangle$, is similar to an NFG, except that the underlying graph is hierarchical. The objective of Player i is to form a path from s_i to t_i in the flat expansion of $\mathcal{G}$. We assume that the objectives of all players are in $\{in_1\} \times Exit_1$, for the entry vertex in_1 and the set $Exit_1$ of exit vertices in the "outer" subgraph G_1. While this strictly restricts the class of games, it is very easy to extend our results to a setting in which the objectives involve arbitrary vertices in $\mathcal{G}$. Essentially, our algorithms proceed from the innermost sub-graph G_n to G_1. The assumption above saves a special treatment for G_1.

We introduce three cost-sharing mechanisms for HNFGs. Consider an HNFG $\mathcal{N} = \langle k, \mathcal{G}, \langle s_i, t_i \rangle_{i \in [k]} \rangle$. Let $\mathcal{G} = \langle G_1, \ldots, G_n \rangle$, with $G_j = \langle V_j, B_j, in_j, Exit_j, \tau_j, E_j, c_j \rangle$. Also, let $\mathcal{N}^f = \langle k, \mathcal{G}^f, \langle s_i, t_i \rangle_{i \in [k]} \rangle$ be the NFG obtained from $\mathcal{N}$ by replacing $\mathcal{G}$ by its flat expansion.

2.3.1 The flat cost-sharing mechanism

In the flat cost-sharing mechanism (Flat-mechanism, for short), the strategies and the costs of the players are defined with respect to $\mathcal{N}^f$. Thus, the only affect of the hierarchical structure in the flat approach is its succinctness. The flat mechanism fits settings in which the traversal of edges corresponds to the formation of physical channels or the utilization of consumable resources. For example, when the network models a hardware design that should be built from a library of components.

Consider, for example, the graph $\mathcal{G} = \langle G_1, G_2 \rangle$ depicted in Figure 1. Let $\mathcal{N} = \langle 2, \mathcal{G}, \{\langle s, t_1 \rangle, \langle s, t_2 \rangle\} \rangle$. Then, the game is played on the flat graph $\mathcal{G}^f$ on the right. Consider the profile $P = \langle \pi_1, \pi_2 \rangle$ in which Player 1 takes the path that traverses both boxes and in both calls to G_2 takes the u_3 exit, and Player 2 takes the path that traverses both boxes and in both calls to G_2 takes the u_4 exit. Then, the players share the edges $\langle s, (b_1, u_1) \rangle$, $\langle (b_1, u_1), (b_1, u_2) \rangle$, and $\langle (b_2, u_1), (b_2, u_2) \rangle$. Accordingly, $cost_1(P) = \frac{2}{2} + \frac{6}{2} + 4 + 4 + 2 + \frac{6}{2} + 4 + 3 = 24$ and $cost_2(P) = \frac{2}{2} + \frac{6}{2} + 5 + 7 + \frac{6}{2} + 5 + 1 = 25$. This is not a stable profile, as Player 1 can reduce his cost to 22 by deviating to the edge $\langle s, t_1 \rangle$. Also, Player 2 can join Player 1 in the first box and reduce his cost to $\frac{2}{2} + \frac{6}{2} + \frac{4}{2} + \frac{4}{2} + \frac{2}{2} + \frac{6}{2} + 5 + 1 = 18$. Note that this deviation also reduces the cost of Player 1, to 19.

2.3.2 The uniform hierarchical cost-sharing mechanism

Recall that $E = \bigcup_{1 \leq j \leq n} E_j$. In the uniform hierarchical (UH) cost-sharing mechanism, a strategy for Player i is a set $\pi_i \subseteq E$ of edges in the hierarchical graph $\mathcal{G}$ such that π_i covers a path from s_i to t_i in $\mathcal{G}^f$. Players' costs in a profile $P = \langle \pi_1, \ldots, \pi_k \rangle$ are defined as follows: For a subgraph G_j and an edge $e \in E_j$, we define the load on e, denoted $load_P(e)$, as the number of strategies in P that include e. Thus, $load_P(e) = |\{i \in [k] : e \in \pi_i\}|$. The cost of an edge is shared evenly by the players that use it. Thus, the cost of Player i in P is $cost_i(P) = \sum_{e \in \pi_i} \frac{c(e)}{load_P(e)}$.

The UH mechanism corresponds to settings in which the traversal of edges corresponds to the utilization of non-consumable resources. Thus, repeated calls to the resource do not require its re-formation. For example, when the network models a software design that should be build from a library of components.

In the uniform sharing rule, we care for the binary information of whether or not a player has used the resource, and we do not distinguish between light and heavy users of the resource.

Consider again the HNFG $\mathcal{N}$, now with the UH mechanism. Let $P = \langle \pi_1, \pi_2 \rangle$ be the profile in which Player 1 takes the path that traverses both boxes and in both calls to G_2 takes the u_3 exit, and Player 2 takes the path that traverses both boxes and in both calls to G_2 takes the u_4 exit. Thus, $\pi_1 = \{\langle s, b_1 \rangle, \langle (b_1, u_3), v \rangle, \langle v, b_2 \rangle, \langle (b_2, u_3), t_1 \rangle, \langle u_1, u_2 \rangle, \langle u_2, u_3 \rangle\}$ and $\pi_2 = \{\langle s, b_1 \rangle, \langle (b_1, u_4), b_2 \rangle, \langle (b_2, u_4), t_2 \rangle, \langle u_1, u_2 \rangle, \langle u_2, u_4 \rangle\}$. The load on $\langle s, b_1 \rangle$ and $\langle u_1, u_2 \rangle$ is 2, and the load on all other edges used in P is 1. Accordingly, $cost_1(P) = \frac{2}{2} + 4 + 2 + 3 + \frac{6}{2} + 4 = 17$ and $cost_2(P) = \frac{2}{2} + 7 + 1 + \frac{6}{2} + 5 = 17$. Now, Player 1 has no incentive to deviate to $\langle s, t_1 \rangle$. However, P is not a NE as Player 2 can join Player 1 in the first box and reduce his cost. Indeed, let $\pi'_2 = \{\langle s, b_1 \rangle, \langle (b_1, u_3), v \rangle, \langle v, b_2 \rangle, \langle (b_2, u_4), t_2 \rangle, \langle u_1, u_2 \rangle, \langle u_2, u_3 \rangle, \langle u_2, u_4 \rangle\}$. Then, in the profile $P' = \langle \pi_1, \pi'_2 \rangle$, we have that $cost_2(P') = \frac{2}{2} + \frac{4}{2} + \frac{2}{2} + 1 + \frac{6}{2} + \frac{4}{2} + 5 = 15$. Note that Player 1 also benefits from this move, as $cost_1(P') = 12$. This example demonstrates that, even-though players have incentive to use an edge multiple times, the optimal strategy of a player in a subgraph G_i need not induce a single path from in_i to some vertex in $Exit_i$. Rather, it is sometimes beneficial for the players to pay for accessing several exit vertices.

2.3.3 The proportional hierarchical cost-sharing mechanism

Like the UH mechanism, the proportional hierarchical (PH) cost-sharing mechanism corresponds to settings in which the traversal of edges corresponds to the utilization of a non-consumable resources. Here, however, we care for the number of times such resources are used by the players, as their costs are proportional to the use. In the PH mechanism, a strategy for Player i is a *multiset* π_i of edges in the hierarchical graph $\mathcal{G}$ such that π_i covers a path from s_i to t_i in $\mathcal{G}^f$. Players' costs in a profile $P = \langle \pi_1, \ldots, \pi_k \rangle$ are defined as follows: For a subgraph G_j and an edge $e \in E_j$, we define the weighted load on e, denoted $wload_P(e)$, as the number of times e appears in all the strategies in P. Recall that for a multiset π, we deonte by $\pi(e)$ the number of times an element e appears in π. Then, $wload_P(e) = \sum_{i \in [k]} \pi_i(e)$, and the cost of Player i in P is $cost_i(P) = \sum_{e \in \pi_i} \frac{\pi_i(e) \cdot c(e)}{wload_P(e)}$.

Back to our example $\mathcal{N}$, the profile P with the PH mechanism consists of the strategies $\pi_1 = \{\langle s, b_1 \rangle^1, \langle u_1, u_2 \rangle^2, \langle u_2, u_3 \rangle^2, \langle (b_1, u_3), v \rangle^1, \langle v, b_2 \rangle^1, \langle (b_2, u_3), t_1 \rangle^1\}$ and $\pi_2 = \{\langle s, b_1 \rangle^1, \langle u_1, u_2 \rangle^2, \langle u_2, u_4 \rangle^2, \langle (b_1, u_4), b_2 \rangle^1, \langle (b_2, u_4), t_2 \rangle^1\}$. Now, $wload_P(\langle s, b_1 \rangle) = wload_P(\langle u_2, u_3 \rangle) = wload_P(\langle u_2, u_4 \rangle) = 2$, $wload_P(\langle u_1, u_2 \rangle) = 4$, and the weighted load on all other edges used in P is 1. Accordingly, every traversal of $\langle u_1, u_2 \rangle$ costs $\frac{6}{4}$, and similarly for the other edges. Hence, $cost_1(P) = \frac{2}{2} + 2 \cdot \frac{6}{4} + 4 + 4 + 2 + 3 = 17$ and $cost_2(P) = \frac{2}{2} + 2 \cdot \frac{6}{4} + 5 + 7 + 1 = 17$. While Player 1 has no incentive to deviate to $\langle s, t \rangle$, Player 2 can reduce his cost by deviating to a path that joins Player 1 in b_1. Indeed, let $\pi'_2 = \{\langle s, b_1 \rangle^1, \langle u_1, u_2 \rangle^2, \langle u_2, u_3 \rangle^1, \langle (b_1, u_3), v \rangle^1, \langle v, b_2 \rangle^1, \langle u_2, u_4 \rangle^1, \langle (b_2, u_4), t_2 \rangle^1\}$. Then, in the profile $P' = \langle \pi_1, \pi'_2 \rangle$, we have that $wload_P(\langle v_1, b_1 \rangle) = 2$, $wload_P(\langle u_1, u_2 \rangle) = 4$, $wload_P(\langle u_2, u_3 \rangle) = 3$, $wload_P(\langle (b_1, u_3), v \rangle) = 2$, $wload_P(\langle v, b_2 \rangle) = 2$, and the weighted load on all other edges used in P is 1. Accordingly, $cost_2(P') = \frac{2}{2} + 2 \cdot \frac{6}{4} + \frac{4}{3} + \frac{4}{2} + \frac{2}{2} + 5 + 1 = 14\frac{1}{3}$. Note that Player 1 also benefits from this move, as $cost_1(P') = \frac{2}{2} + 2 \cdot \frac{6}{4} + 2 \cdot \frac{4}{3} + \frac{4}{2} + \frac{2}{2} + 3 = 12\frac{2}{3}$.

3 Stability Existence and Inefficiency

In this section we study the stability of HNFGs. We show that the cost-sharing mechanism is crucial in this analysis. Specifically, HNFGs with the Flat or the UH mechanism have an NE and their PoA and PoS are identical to the bounds known for NFGs. On the other hand, we show that even simple instances of HNFGs with the PH mechanism need not have an NE, and there are games for which the only stable profile is k times more expensive than the SO.

We start with the stability existence question. The proof of the following theorem is based on converting every HNFG with the flat or the UH mechanism to an equivalent resource-allocation game, which is known to have an NE. As we show, the relation with resource-allocation games also induces potential functions for HNFGs in the flat and UH mechanisms.

Theorem 3.1 *Every HNFG with the flat or UH mechanism has an NE.*

Proof: An HNFG $\mathcal{N}$ with the flat mechanism is identical by definition to an NFG played on $\mathcal{N}^f$. For the UH mechanism we refer to the generalization of NFGs to resource-allocation cost-sharing games, in which players' strategies are subset of a given set E of resources. Unlike NFGs, in which a strategy for player i is a set of resources (that is, edges) that form an (s_i, t_i)-path, in general resource-allocation games, the strategies are arbitrary subsets of E.

Let $\mathcal{N} = \langle k, \mathcal{G}, \langle s_i, t_i \rangle_{i \in [k]} \rangle$. By taking the set of edges in $\mathcal{G}$ to be the set of resources, and defining the set of strategies for Player i as the set of subsets of edges that cover a path from s_i to t_i, we can reduce $\mathcal{N}$ with the UH mechanism to a resource allocation game. Note that listing all these strategies may be exponential in $|\mathcal{G}|$. Here however, we only care about NE existence and not about its calculation.

It is well-known that every resource-allocation cost-sharing game has a NE. Moreover, let $\Phi(P) = \sum_{e \in E} c(e) \cdot H(load_P(e))$, where $H(0) = 0$, and $H(k) = 1 + 1/2 + \ldots + 1/k$, then, as shown in [8], $\Phi(P)$ is a potential function whose value reduces with every beneficial step of a player. This implies that every best-response sequence is guaranteed to converge to an NE. ■

For the PH mechanism, we present a negative result.

Theorem 3.2 *An HNFG with the PH mechanism need not have an NE.*

Proof: Consider the hierarchical graph $\mathcal{G} = \langle G_1, G_a, G_b, G_c \rangle$ depicted in Figure 2. Let $\mathcal{N} = \langle 2, \mathcal{G}, \{\langle s, t_i \rangle\}_{i \in \{1,2\}} \rangle$. For every $\sigma \in \{a, b, c\}$, we have that $\tau_1(b_\sigma) = G_\sigma$. In the figure, edges that are not labeled are free. Thus, Player 1 needs to select in G_1 one of the two paths $\rho_1^1 = (s, b_a, b_c, t_1)$ and $\rho_1^2 = (s, b_b, t_1)$, and Player 2 needs to select one of two paths $\rho_2^1 = (s, b_a, b_a, b_a, t_2)$ and $\rho_2^2 = (s, b_c, b_a, b_a, t_2)$.

We show that $\mathcal{N}$ with the PH mechanism does not have an NE. Recall that a strategy for Player i is a multiset π_i over edges in $\mathcal{G}$ such that π_i covers a path from s to t_i in $\mathcal{G}^f$. Since all the edges in E_1 are free, we describe the players' strategies as multisets that include only the edges in the subgraphs G_a, G_b, and G_c. Denote by e_a, e_b, and e_c the (only) edge in G_a, G_b and G_c respectively. Thus, for Player 1, we have strategies $\pi_1^1 = \{e_a, e_c\}$ and $\pi_1^2 = \{e_b\}$, and for Player 2 we have $\pi_2^1 = \{e_a, e_a, e_a\}$ and $\pi_2^2 = \{e_c, e_a, e_a\}$.

Table 1 describes the players' costs in the four possible profiles. Note that $c(e_a) = 36, c(e_b) = 12$ and $c(e_c) = 2$. Consider for example the top left profile $P = \langle \pi_1^1, \pi_2^1 \rangle$. In this profile, the edge e_a is

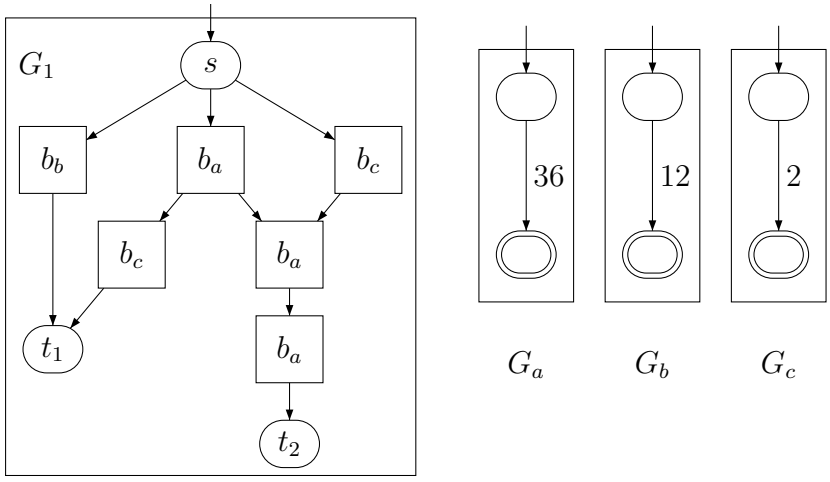

Figure 2: An HNFG that has no NE with the PH mechanism.

traversed four times, e_b is not traversed at all, and e_c is traversed once. Thus, $wload_P(e_a) = 4$. This implies that every traversal of e_a costs $c(e_a)/wload_P(e_a) = 36/4 = 9$. Since $wload_P(e_c) = 1$ and $e_c \in \pi_1^1$, Player 1 should also cover the cost of e_c. Hence, $cost_1(P) = 9 + 2 = 11$ and $cost_2(P) = 3 \cdot 9 = 27$. The players' costs in all other profiles are calculated in a similar way. The costs in the table imply that players benefit from changing strategies in counter clockwise direction, thus no NE exists.

	$\{e_a, e_c\}$	$\{e_b\}$
$\{e_a, e_a, e_a\}$	$11, 27$	$12, 36$
$\{e_c, e_a, e_a\}$	$13, 25$	$12, 38$

Table 1: Players' costs in $\mathcal{N}$. Each entry describes the cost of Player 1 followed by the cost of Player 2.

■

We turn to analyze the equilibrium inefficiency. Once again, the fact that each HNFG with the flat or the UH mechanism has an equivalent resource-allocation cost-sharing game enables us to adopt the upper bounds known for resource-allocation games to our setting. Matching lower bounds then follow from the known bounds on NFGs and the fact that every NFG can be viewed as an HNFG with no nesting of subgraphs.

Theorem 3.3 *The* PoS *and* PoA *of k-player HNFGs with the flat or the UH mechanism are $O(\log k)$ and k, respectively.*

For the PH mechanism, we show that stability may came with a high cost, strictly higher than the one known for NFGs.

Theorem 3.4 *The* PoS *and* PoA *of k-player HNFGs with the PH mechanism are k.*

Proof: Similar to the analysis of many other cost-sharing games, PoA $\leq k$ as otherwise, some player in some NE profile P is paying more than the SO, and can benefit from deviating to his strategy in the

SO, contradicting the stability of P. Combining the fact that PoA $\geq$ PoS, it is sufficient to show that PoS $\geq k$ in order to establish the tight bounds.

For every $k > 1$, we describe a k-player HNFG $\mathcal{N}_k$ such that the cost of the only NE in $\mathcal{N}_k$ is kM, for some large constant M, whereas the SO is $M + \epsilon''$, for a small constant ϵ''. Assume first that k is even. Partition the set $[k]$ of players into pairs $\langle 2\ell - 1, 2\ell \rangle$ for $1 \leq \ell \leq \frac{k}{2}$. Let $\mathcal{N}^\ell$ be a 2-player HNFG with no NE, with the costs of its edges multiplied by a small constant ϵ. In particular, we refer to the HNFG described in the proof of Theorem 3.2.

The HNFG $\mathcal{N}_k$ is played on the hierarchical graph $\mathcal{G} = \langle G_0, \{G_1^\ell, G_a^\ell, G_b^\ell, G_c^\ell\}_{1 \leq \ell \leq k/2} \rangle$, where G_0 is depicted in Figure 3, and the other components consists of $k/2$ copies of the graphs G_1, G_a, G_b, and G_c, described in Figure 2, with all costs multiplied by ϵ. The graph G_0 includes an edge $\langle s, t \rangle$ of cost kM, an edge $\langle s, v \rangle$ of cost M, and $k/2$ free edges $\langle v, s_\ell \rangle$ leading the copies G_1^ℓ for $1 \leq \ell \leq \frac{k}{2}$.

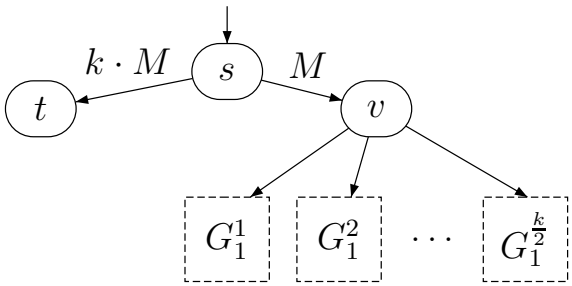

Figure 3: An HNFG $\mathcal{N}_k$ for which $\mathrm{PoS} = k$. Every G_1^ℓ is a copy of G_1 depicted in Figure 2.

For simplicity, we assume that each player can choose between one of two targets. It is easy to see that this assumption can be removed by adding a new target connected from the two targets by free edges. Consider the ℓ-th pair of players. The target vertices of the first player in the pair are t and t_1^ℓ. The target vertices of the second player are t and t_2^ℓ. Thus, every player has three strategies: the path consisting of the edge $\langle s, t \rangle$ and the paths starting with s, v, s_ℓ and continuing with one of the two strategies in G_1^ℓ, as detailed in the proof of Theorem 3.2.

The SO of $\mathcal{N}_k$ consists of edges from the right side of the network: the edges $\langle s, v \rangle$, $\{\langle v, s_\ell \rangle_{1 \leq \ell \leq k/2}\}$, and edges forming an SO for each of the disjoint $\frac{k}{2}$ games (the latter consists of the edges e_a^ℓ, e_c^ℓ, and additional free edges from G_1^ℓ). The cost of the SO is then $M + \frac{\epsilon' k}{2}$, for $\epsilon' = 38\epsilon$.

We show that the only NE in $\mathcal{N}_k$ is the profile in which all the players share the edge $\langle s, t \rangle$. This profile is indeed an NE, as the cost of every player is exactly M, and by deviating to the right side of the network, a player must pay the cost of $\langle s, v \rangle$ plus the cost of his chosen path in some G_1^ℓ, which together exceeds M. Moreover, this is the only NE since in every other profile, players would benefit from leaving the edge $\langle s, t \rangle$ and reaching $\mathcal{N}^\ell$ – our familiar no-NE game described in the proof of Theorem 3.2. The cost of this NE profile is kM, implying that the PoS is $\frac{Mk}{M+19k\epsilon}$, which tends to k.

Finally, if the number k of players is odd, we define for the unpaired player two strategies: one is the path $\langle s, t \rangle$, and the other is a path s, v, u for a new vertex u. By setting to ϵ the cost of $\langle u, v \rangle$, it still holds that $\langle s, t \rangle$ is the only NE profile. The PoS for an odd k is therefore $\frac{Mk}{M+(19k+1)\epsilon}$, which tends to k. ∎

4 Computational Complexity

In this section we study the complexity of reasoning about HNFGs in the different cost-sharing mechanisms. The principal question is whether the exponential succinctness of HNFGs leads to an exponential increase in the complexity of reasoning about them.

4.1 The UH and PH mechanisms

Recall that a strategy for Player i in the UH or PH mechanism is a set or a multiset π_i over E. A strategy is *feasible* if there is a path ρ from s_i to t_i in $\mathcal{G}^f$ such that ρ is covered by π_i. In traditional NFGs, it is easy to check in polynomial time whether a given set of edges is a feasible strategy. Indeed, there, the underlying graph is given explicitly. This is not the case in HNFGs: given π_i, a naive check that π_i indeed covers a path from s_i to t_i in $\mathcal{G}^f$ involves a construction of $\mathcal{G}^f$, which may be exponential in $\mathcal{G}$. An efficient checking that a given strategy π_i is feasible requires a clever encoding of π_i, involving a restriction to a subset of all possible strategies. We first define this subset and prove that it is dominating, that is, every Player has a best-response move to a homogeneous strategy.

Recall that π_i is feasible if there is a path ρ from s_i to t_i in $\mathcal{G}^f$ such that ρ is covered by π_i. The path ρ may traverse subgraphs G_j of $\mathcal{G}$ several times (in fact, per Observation 2.1, even exponentially many times). In each traversal, the path ρ may exit G_j through different exit vertices. For example, in the HNFGs described in Section 2.3, we showed that the players benefit from taking a strategy that exits G_2 from both u_3 and u_4. Thus, restricting attention to strategies in which all the traversals of G_j use the same exit vertex is not sound (and in fact may affect not only the cost of Player i but also cause t_i not to be reachable from s_i). Consider now two traversals of the subgraph G_j in which Player i chooses to exit G_j through the same exit vertex $u \in \mathit{Exit}_j$. Here too, Player i may choose to fulfill this repeated "nested sub-objective" in different ways. We say that a strategy for Player i is *homogeneous* if for every $j \in [n]$ and every $u \in \mathit{Exit}_j$, whenever Player i traverses the subgraph G_j through exit u it uses the same $\langle in_j, u\rangle$-path. We claim that restricting attention to homogeneous strategies is sound, and also leads to an efficient feasibility check:

Lemma 4.1 *Consider an HNFG $\mathcal{N}$ with the UH or PH mechanism, and a player $i \in [k]$.*

1. *Every non-homogeneous strategy for Player i is dominated by a homogeneous one.*
2. *Checking that a homogeneous strategy of Player i is feasible can be done in poly-time.*

Proof: We start with the first claim. We show that as long as Player i follows a non-homogeneous strategy, he can reduce the degree of non-homogeneousity while only reducing his cost. Repeating this process, which is guaranteed to terminate, results in a homogeneous strategy.

Let P be a profile in which Player i crosses G_j through exit u more than once, and it uses two different paths, p_1 and p_2. In the UH mechanism, p_1 and p_2 are sets of edges, and Player i pays for using $p_1 \cup p_2$ regardless of the number of times he traverses each edge in the set. In this case, Player i can clearly only benefit from using, say, p_1 in both crosses.

In the PH mechanism, p_1 and p_2 are mulisets of edges, and the analysis is more involved. By the definition of the PH mechanism, for a single use of p_l, for $l \in \{1, 2\}$, Player i pays $\sum_{e \in p_l} \frac{c(e)p_l(e)}{wload_P(e)}$.

Player i would benefit from switching from p_2 to p_1 iff the cost he saves from leaving p_2 in P is higher than the added cost experienced from joining p_1; that is, iff

$$(*) \quad \sum_{e \in p_2 \setminus p_1} \frac{c(e)(p_2 \setminus p_1)(e)}{wload_P(e)} > \sum_{e \in p_1 \setminus p_2} \frac{c(e)(p_1 \setminus p_2)(e)}{wload_P(e) + (p_1 \setminus p_2)(e)}.$$

If $(*)$ holds, Player i's cost in P can be reduced by following p_1 in both traversals, and we are done. Otherwise, we show that switching from p_1 to p_2 is beneficial. Indeed,

$$\sum_{e \in p_1 \setminus p_2} \frac{c(e)(p_1 \setminus p_2)(e)}{wload_P(e)} > \sum_{e \in p_1 \setminus p_2} \frac{c(e)(p_1 \setminus p_2)(e)}{wload_P(e) + (p_1 \setminus p_2)(e)} \geq$$

$$\geq \sum_{e \in p_2 \setminus p_1} \frac{c(e)(p_2 \setminus p_1)(e)}{wload_P(e)} > \sum_{e \in p_2 \setminus p_1} \frac{c(e)(p_2 \setminus p_1)(e)}{wload_P(e) + (p_2 \setminus p_1)(e)}.$$

For the second claim, checking the feasibility of homogeneous strategies requires only one check for each subgraph G_j and exit vertex $u \in Exit_j$, which can be done in polynomial time. ■

We proceed to study the complexity of finding a BR and an SO in HNFGs with the UH or PH mechanism. For NFGs, a BR move can be found in polynomial time, and the problem of finding an SO is NP-complete [32]. For the lower bound, we show two reductions, both with a single-player HNFG. One, for the case the depth of the HNFG is a constant, is from the *directed Steiner tree* problem; and one, for the case the number of exit vertices is a constant, is from the *hitting-set* problem.

Theorem 4.2 *The problem of finding a BR move for a HFNG with the UH or PH mechanism is NP-complete. NP-hardness holds already for single-player HNFGs of a constant depth or with a constant number of exit vertices.*

Proof: We start with the upper bound. Given an HNFG $\mathcal{N}$, a profile P for $\mathcal{N}$, a threshold $c \in$, and $i \in [k]$, a witness for a BR move is a strategy π_i for which $cost_i([P_{-i}, \pi_i]) \leq c$. By Lemma 4.1, the strategy π_i can be homogeneous, which implies it can be checked in polynomial time.[3]

For the lower bound, we show two reductions, both with a single-player HNFG. One for the case the depth of the HNFG is a constant, and one for the case the number of exit vertices is a constant. We start with the first case and describe a reduction from the *directed Steiner tree* problem. There, we are given a directed weighted graph $G = \langle V, E, c \rangle$, a source vertex $s \in V$, and a set of terminal vertices $T \subseteq V$, and we seek a tree $T \subseteq E$ that connects s to all the vertices in T and is of minimal cost. Given an input $\langle G, s, T \rangle$ to the directed Steiner tree problem, we describe an HNFG $\mathcal{N}$ with a single player such that an optimal strategy for the player corresponds to an optimal Steiner tree in $\langle G, s, T \rangle$. Let $T = \{t_1, \ldots, t_l\}$. The network $\mathcal{N}$ is defined with a hierarchical graph $\mathcal{G} = \langle G_1, G_2 \rangle$, for the subgraphs G_1 and G_2 appearing in Figure 4. The subgraph G_1 consists of a chain of l boxes. All boxes call the subgraph G_2, which is copy of G with the vertices in T being exit-vertices (for clarity, in the figure we omit the edges in G and their costs). The chain starts with the vertex v_0 and ends with t, and the objective of the player is $\langle v_0, t \rangle$. In the i-th box, we exit from t_i to the $(i+1)$-th box. In the last box,

[3]Note that we handle here the decision-problem variant of the problem, namely one with a threshold $c \in \mathbb{R}_{\geq 0}$. The optimization variant then searches for a witness in the optimal threshold.

we exit from t_l to t. The cost of the edges in G_2 coincides with their cost in G. The edges in G_1 are free.

By the construction of $\mathcal{G}$, a strategy π for the player has to cover a path that includes paths from s to t_i, for all $t_i \in T$. Accordingly, π corresponds to a tree in G that connects s to all the vertices in T. Recall that in the UH mechanism, the cost of π is the sum of the costs of its edges. Since $\mathcal{N}$ has a single player, this coincides with the cost of the induced Steiner tree. The fact there is only one player also implies that the same reasoning holds for the PH mechanism. Indeed, in both cases the cost of the player is the cost of the profile, which is the sum of the costs of the edges in G_2 that are used, namely the cost of the Steiner tree.

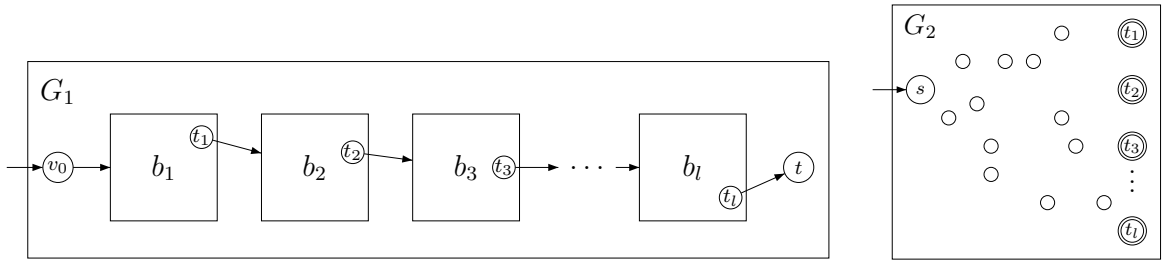

Figure 4: A reduction from the directed Steiner tree problem.

For the second case, namely when the number of exit vertices is a constant, we describe a reduction from the *hitting-set* problem. An instance of the hitting-set problem consists of a set C of subsets of a finite set S. The goal is to find a subset $S' \subseteq S$ of minimum cardinality, such that S' contains at least one element from each subset in C.

Given S and $C = \{C_1, \ldots, C_\ell\}$, we construct an HNFG $\mathcal{N}$ with a single player such that an optimal strategy for the player corresponds to a subset S' of minimal cardinality such that $S' \cap C_i \neq \emptyset$ for all $i \in [\ell]$. The graph $\mathcal{G}$ on top of which $\mathcal{N}$ is defined consists of an outer graph G_1 and $|S|$ additional subgraphs (See an example in Figure 5). For every element $j \in S$, the subgraph G_j has no boxes and consists of one directed edge whose cost is 1. Assume that $C = \{C_1, C_2, \ldots, C_\ell\}$. The outer graph G_1 consists of ℓ layers, each corresponding to a set in C. The vertices of G_1 are $\{v_0, v_1, \ldots, v_\ell\}$, and its boxes are mapped to the subgraphs $\{G_j\}_{j \in S}$. The i-th layer of G_1 connects v_{i-1} and v_i by $|C_i|$ parallel 2-edge paths: v_{k-1}, b_j, v_k, for all $j \in C_k$. All the edges in G_1 are free.

Consider a player whose objective is (v_0, v_ℓ). Every path connecting these edges must cross all the layers. It is easy to see that crossing the i-th layer involves traversing a box b_j for some $j \in C_i$. The cost of a path is exactly the number of different subgraphs called by boxes along the path. Thus, a hitting set of minimum cardinality corresponds to a strategy with minimum cost. Since $\mathcal{N}$ has a single player, this holds for both the UH and PH mechanisms. ∎

Thus, the exponential succinctness of HNFGs makes the BR problem for the UH and PH mechanisms exponentially more complex than the one for NFGs. Since the BR problem in single-player HNFGs coincides with the SO problem, Theorem 4.2 immediately implies the lower bound in the following theorem. The upper bound follows from the fact that a witness to the SO consists of listing the set of edges purchased in every subgraph. It is easy to see that there exists an SO in which every player is assigned a homogeneous strategy, therefore, the SO's feasibility is tractable.

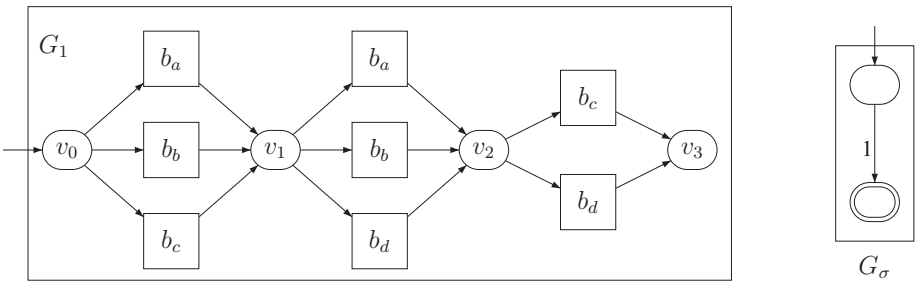

Figure 5: A reduction from the hitting-set problem, for $S = \{a, b, c, d, e\}, C = \{\{a, b, c\}, \{a, b, d\}, \{c, d\}\}$.

Theorem 4.3 *The problem of finding an SO for an HNFG with the UH or PH mechanism is NP-complete. NP-hardness holds already for single-player HNFGs of a constant depth or with a constant number of exit vertices.*

Recall that an HNFG with the PH mechanism need not have an NE (Theorem 3.2). A natural question arising from this result is whether we can distinguish between instances that have or do not have a stable profile. We show the following:

Theorem 4.4 *Deciding whether an NE exists in an HNFG with the PH mechanism is in Σ_P^2 and is NP-hard.*

Proof: The class Σ_2^P contains all problems that can be solved by a nondeterministic polynomial-time Turing machine with an oracle to an NP problem. Given an HNFG with the PH mechanism, the problem of deciding whether an NE exists is in Σ_2^P: A nondeterministic Turing machine can guess an NE and then use polynomially many (one for each player) calls to an oracle that checks that no player has an incentive to deviate. Since the latter problem for HNFGs is in NP (Theorem 4.2), membership in Σ_2^P follows.

We now prove that the problem is NP-hard. The idea of the hardness proof is to extend the no-NE game from the proof of Theorem 3.2 such that one of the players has additional strategies. In particular, he would prefer to avoid the two non-stable strategies if a given instance of an NP decision problem $\mathcal{P}$ has a positive solution. This way, deciding whether an NE exists amounts to solving $\mathcal{P}$.

Specifically, we describe a reduction from the *hitting-set* problem (defined in the proof of Theorem 4.2). In the decision problem, we are given an integer k, and the goal is to decide whether a hitting-set of size k exists. We modify the no-NE game from the proof of Theorem 3.2 as follows. In our modified HNFG $\mathcal{N}'$, Player 1 has additional strategies given by the gadget corresponding to the hitting-set problem, as described in the proof of Theorem 4.2. Using these strategies, Player 1 can select an $\langle s, t_1 \rangle$-path that crosses the hitting-set gadget. We set the cost of the single edge in each of the subgraphs $\{G_j\}_{j \in S}$ to $2 + \frac{1-\epsilon}{k}$. Also, the edge (s, b_b) in G_1 (see Figure 2) has cost $2k - 10$. Thus, if Player 1 chooses a path in the hitting-set gadget corresponding to a hitting-set of size k', his cost would be $k'(2 + \frac{1-\epsilon}{k})$, and if he chooses a path in the original graph G_1, his cost would be as specified in Table 1, plus the cost of (s, b_b). Table 2 gives the Players' costs is each profile.

Assume that a hitting-set of size k exists. Player 1 can choose the path corresponding to the hitting set

	$\{e_a, e_c\}$	$\{e_b\}$	k'-hitting-set path
$\{e_a, e_a, e_a\}$	$11 + (2k - 10) = 2k + 1, 27$	$12 + (2k - 10) = 2k + 2, 36$	$k'(2 + \frac{1-\epsilon}{k}), 36$
$\{e_c, e_a, e_a\}$	$13 + (2k - 10) = 2k + 3, 25$	$12 + (2k - 10) = 2k + 2, 38$	$k'(2 + \frac{1-\epsilon}{k}), 38$

Table 2. Players' costs in $\mathcal{N}'$ with the PH mechanism. Each entry describes the cost of Player 1 followed by the cost of Player 2.

and pay $2k + 1 - \epsilon$. It is easy to verify that the top-right profile in the table is an NE. On the other hand, if the smallest hitting-set has size at least $k + 1$, then every gadget-path has cost at least $2k + 3$ (for $\epsilon < \frac{1}{k}$), implying that Player 1 never selects a gadget-path, which leaves us with the original no-NE game. We conclude that an NE exists if and only if the answer to the hitting-set decision problem is positive. ∎

4.2 The flat mechanism with a constant number of exit vertices

Consider an HNFG played over a hierarchical graph $\mathcal{G}$. Recall that in the flat mechanism, costs are calculated with respect to $\mathcal{G}^f$, which is exponentially larger than $\mathcal{G}$. While the exponential blow-up applies already for hierarchical graphs in which the number of exit vertices in each subgraph is a constant (in fact, per Observation 2.1, is 1), experience in formal verification of hierarchical systems shows that reasoning about hierarchical-FSMs in which each subgraph has a constant number of exit vertices does make verification easier [6, 7]. In this section we consider HNFGs that are played over hierarchical graphs in which each subgraph has a constant number of exit vertices. We denote this class by CE-HNFGs. We note that CE-HNFGs are common in practice: in software, procedures typically have a constant number of returns, and in hardware, nested boxes are plugged in via a constant number of connections.

Before we describe our results for CE-HNFGs, let us point out that there are additional aspects in which the flat mechanism is computationally easier than the UH and PH mechanisms. For example, while the problem of finding an SO in HNFGs in the UH or PH mechanism is NP-complete already for single-player CE-HNFGs (by Theorem 4.3), for the flat mechanism, the single-player instance is easy even without restricting to CE-HNFGs. Indeed, let $\mathcal{N} = \langle 1, \mathcal{G}, \langle s, t\rangle\rangle$, with $\mathcal{G} = \langle G_1, \ldots, G_n\rangle$. Starting with G_n, we recursively replace each box that calls a subgraph G_j by a tree of depth 1 with root in_j and edges to all exit vertices $t \in Exit_j$. The cost of such an edge is the cost of the shortest path from in_j to t, which we need to calculate only once (and after boxes in G_j have been recursively replaced by trees of depth 1). Thus, we have,

Theorem 4.5 *The problem of finding an SO in a single-player HNFG with the flat mechanism can be solved in polynomial time.*

For $k > 1$ players, finding an SO is still tractable, but the algorithm is more involved:

Theorem 4.6 *The problem of finding an SO in CE-HNFGs with the flat mechanism can be solved in polynomial time.*

Proof: Let $\mathcal{N}$ be a CE-HNFG with $\mathcal{G} = \langle G_1, \ldots, G_n\rangle$, where $G_j = \langle V_j, B_j, in_j, Exit_j, \tau_j, E_j, c_j\rangle$. A profile of $\mathcal{N}$ utilizes a subset of the edges in $\mathcal{G}$. In fact, for every box in $\mathcal{G}$ that calls a subgraph

G_j, the utilized edges form a Steiner tree connecting in_j with a set $T \subseteq Exit_j$ of exit vertices. Our algorithm is based on the fact that these Steiner trees can be enumerated, and that the minimum Steiner tree problem can be solved efficiently when the number of terminals is a constant [25].

For an index $j \in [n]$ and a set $T \subseteq Exit_j$, we define the HNFG $\mathcal{N}_{j,T} = \langle |T|, \mathcal{G}_j, \langle in_j, t\rangle_{t\in T}\rangle$, where $\mathcal{G}_j = \langle G_j, G_{j+1}, \ldots, G_n\rangle$. That is, $\mathcal{N}_{j,T}$ is a $|T|$-player game, where each player tries to reach from in_j to a different exit vertex $t \in T$. Note that an SO in $\mathcal{N}_{j,T}$ is a profile that minimizes the cost required for forming paths from in_j to all vertices in T in the flat expansion of $\mathcal{G}_j$. Now, let G'_j be a weighted tree of depth 1 with root in_j and leaves in 2^{Exit_j}, where the cost of an edge $\langle in_j, T\rangle$, for $T \subseteq Exit_j$, is the SO in $\mathcal{N}_{j,T}$. Thus, G'_j describes, for every subset $T \subseteq Exit_j$, the cost of covering paths from in_j to all vertices in T in the flat expansion of $\mathcal{G}_j$. Note that since $|Exit_j|$ is constant, so is the size of G'_j.

We argue that for all $j \in [n]$ and $T \subseteq Exit_j$, there is an algorithm that finds an SO in $\mathcal{N}_{j,T}$ and constructs G'_j in polynomial time. In particular, taking $j = 1$ and $T = \cup_{i\in[k]}\{t_i\}$, we can find the SO of $\mathcal{N}$ in polynomial time.

The algorithm proceeds from the innermost subgraphs, and replaces boxes that call a subgraph G_j by the weighted tree G'_j. In more details, we start with G_n, and construct G'_n in polynomial time. Since G_n does not have boxes, this involves the calculation of the SO in $2^{|Exit_n|}$ NFGs of size $|G_n|$; that is, the games $\mathcal{N}_{n,T}$, for $T \subseteq Exit_n$. Finding the SO in each of these games corresponds to solving a minimum Steiner tree problem with a constant number of terminals, that can be solved in polynomial time [25]. Thus, edges' weights in G'_n can be calculated in polynomial time.

Consider now an index $j \in [n]$ and assume we have already constructed G'_h for all $h > j$. We construct G'_j as follows. First, we remove all boxes from G_j. We do this by replacing each box $b \in B_j$ for which $\tau_j(b) = h$ by a copy of G'_h. The copy is connected to the other vertices of G_j in the following way. First, edges with target b, are redirected to in_h. Then, a vertex $T \subseteq Exit_h$ in G'_h has edges to all vertices that are reachable in G_j by an edge with source (b, t), for some $t \in T$. Intuitively, taking the edge $\langle in_h, T\rangle$ in this copy of G'_h amounts to forming paths to all the exit vertices in T in the flat expansion of G_h that is called from the box b. Accordingly, once we pay the cost of traversing $\langle in_h, T\rangle$ in this copy, we can continue from all the vertices in T. Replacing a box by a copy of G'_h increases the size of G_j by constantly many vertices. Thus, removing all boxes in G_j increases the number of vertices by $O(|B_j|)$. We can now construct G'_j by finding the SO in $2^{|Exit_j|}$ NFGs of size $O(|G_j|)$, which can be done in polynomial time. ∎

We turn to the problem of calculating an NE. A well-known approach for calculating an NE in NFGs is *best-response dynamics* (BRD): starting with an arbitrary profile, we let players perform BR moves until an NE is reached. The complexity class PLS contains local search problems with polynomial time searchable neighborhoods [24]. Essentially, a problem is in PLS if there is a set of feasible solutions for it such that it is possible to find, in polynomial time, an initial feasible solution and then iteratively improve it, with each improvement being performed in polynomial time, until a local optimum is reached. While every iteration of BRD takes polynomial time, the number of iterations need not be polynomial. The problem of finding an NE in NFGs is known to be PLS-complete. We show how to implement BRD in CE-HNFGs in a way that keeps the polynomial time-complexity for each improvement step. The idea is to use a succinct representation of a profile in a CE-HNFG, and to restrict attention to a limited class of profiles that are guaranteed to include an NE.

Theorem 4.7 *The problem of finding an NE in CE-HNFGs with the flat mechanism is PLS-complete.*

Proof: Hardness in PLS follows from hardness for NFGs. We prove membership in PLS. Let $\mathcal{N} = \langle k, \mathcal{G}, \langle s_i, t_i \rangle_{i \in [k]} \rangle$ be a CE-HNFG. Consider the subgraph G_j. In every profile of the game, every copy of G_j may be traversed by at most k players. Each of these players traverses a $\langle in_j, u \rangle$-path for some $u \in Exit_j$. Let $z_j = |Exit_j|$ and let $Exit_j = \{u_1, u_2, \ldots, u_{z_j}\}$. A vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{z_j})$ of integers such that $\sum_\ell \alpha_\ell \leq k$ describes the demand of each of the exit vertices of G_j when it is traversed by $k_\alpha = \sum_{\ell \in [z_j]} \alpha_\ell$ players. Intuitively, it indicates that for all $\ell \in [z_j]$, there are α_ℓ players that seek to exit via u_ℓ. Consider the game $\mathcal{N}_j^\alpha = \langle k_\alpha, \mathcal{G}_j, \langle in_j, u_i \rangle_{i \in [k_\alpha]} \rangle$, where $\mathcal{G}_j = \langle G_j, \ldots, G_n \rangle$. Intuitively, $\mathcal{N}_j^\alpha$ models k_α players that traverse G_j with demand vector α. Note that for every $j \in [n]$, the number of different games demand vectors for G_j, namely the number of different vectors α of z_j positive integers that sum up to an integer bounded by k, is $O(k^{z_j})$.

We claim that finding an NE for $\mathcal{N}_j^\alpha$ is in PLS. We say that a profile for $\mathcal{N}_j^\alpha$ is *fair* if for every exit vertex $u \in Exit_j$, all the players that traverse G_j through a $\langle in_j, u \rangle$-path use the same path.

We first show that we can restrict attention to fair profiles, that is, for every exit vertex $u \in Exit_j$, all the players that traverse G_j through a $\langle in_j, u \rangle$-path use the same path. We claim that every NE of an HNFG with the flat mechanism is fair. The proof is by contradiction: Let P be a profile in which Players a and b use different paths when crossing G_j through a $\langle in_j, u \rangle$-path. We show that P cannot be an NE. Assume w.l.o.g. that $cost_a(P) \leq cost_b(P)$. Assume that Player b joins Player a's $\langle in_j, u \rangle$-path. The load on every edge along this path remains the same or increases. Also, since they used different paths, for at least one edge the load is increased. Thus, this deviation strictly reduces Player b's cost.

Our algorithm for calculating an NE in $\mathcal{N}$ proceeds by calculating for every $j \in \{n, n-1, \ldots, 2\}$ the NEs for all games $\mathcal{N}_j^\alpha$ that can be played on the flat expansion of G_j by a subset of at most k players. By the above, we can restrict attention to fair NEs, which can be described in space $O(|G_j|)$.

The algorithm begins with G_n. Since G_n has no boxes, calculating an NE for every demand vector can be done by BRD. Assume we have already calculated NEs for the games $\mathcal{N}_h^\alpha$ for all $h > j$ and feasible demand vectors α. We describe how to calculate NEs for games over G_j. Consider a demand vector α for G_j and the HFNG $\mathcal{N}_j^\alpha = \langle k_\alpha, \mathcal{G}_j, \langle in_j, u_i \rangle_{i \in [k_\alpha]} \rangle$. We show how to calculate an NE for $\mathcal{N}_j^\alpha$. Initially, we pick an arbitrary $\langle in_j, u_\ell \rangle$-path for all the α_ℓ players whose objective is to reach u_ℓ. The union of these paths determines for every box in G_j how many players traverse it and through which exit. Thus, every box b defines a sub-game $\mathcal{N}_{j'}^{\alpha'}$ for $j' = \tau_j(b)$ and vector α' as determined by the current profile. Since $j' > j$, the game $\mathcal{N}_{j'}^{\alpha'}$ has already been analyzed and the box can be replaced by a gadget with only a source $in_{j'}$ and exit vertices for every $u \in Exit_{j'}$. The cost of the edge $(in_{j'}, u)$ is the cost for one player in the NE calculated for $\mathcal{N}_{j'}^{\alpha'}$.

In order to calculate a BR move in $\mathcal{N}_j^\alpha$ for a specific player $i \in [k]$, we need to solve a shortest path problem in G_j in which edge weights correspond to the cost to be charged if Player i keeps using or joins this edge. For edges of type Top, Call, and Return, this is simply the edge cost divided by the current number of players using it (the load in the given NE profile) plus one if Player i does not currently use it. For an Internal edge (v, u) that calls the subgraph $G_{j'}$, if Player i currently uses this edge, then his cost remains as in the NE calculated for $\mathcal{N}_{j'}^{\alpha'}$. Otherwise, we need to calculate the cost Player i would experience if he moves to a path that includes this edge. Let α' be the demand vector of length $z_{j'}$ describing the demand for exit vertices in $G_{j'}$ in the current profile. Let α'' be the demand vector obtained from α by increasing by one the entry corresponding to the exit vertex u. The game

$\mathcal{N}_{j'}^{\alpha''}$ has been analyzed already, and in particular, it is known what the cost of a player traversing to exit u is. Indeed, this cost is the cost of the Internal edge (v, u) in $\mathcal{N}_j^{\alpha}$.

Let $z = \max_{j \in [n]} z_j$. Using the above algorithm, it is possible to calculate an NE for each of the $O(k^z)$ games $\mathcal{N}_j^{\alpha}$ for $j \in \{n, n-1, \ldots, 2\}$. Once we are done computing a set of NEs for games that can be played on the flat expansion of G_j, for $j \in \{n, n-1, \ldots, 2\}$, we can calculate using the same method an NE for our instance $\mathcal{N} = \mathcal{N}_1^{\alpha}$ for the vector α corresponding to $\mathcal{N}$. ■

References

[1] S. Albers, S. Elits, E. Even-Dar, Y. Mansour, and L. Roditty. On Nash equilibria for a network creation game. In *Proc. 7th SODA*, pages 89–98, 2006.

[2] S. Almagor, G. Avni, and O. Kupferman. Repairing multi-player games, In *Proc. 26th CONCUR*, pages 325–339, 2015.

[3] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[4] G. Avni and O. Kupferman. Synthesis from component libraries with costs. In *Proc. 25th CONCUR*, LNCS 8704, pages 156–172, 2014.

[5] R. Alur, S. Kannan, and M. Yannakakis. Communicating hierarchical state machines. In *Proc. 26th ICALP*, LNCS 1644, pages 169–178. Springer, 1999.

[6] R. Alur and M. Yannakakis. Model checking of hierarchical state machines. *ACM TOPLAS*, 23(3):273–303, 2001.

[7] B. Aminof, O. Kupferman and A. Murano, Improved model checking of hierarchical systems, In *Information and Computation*. 210, pages 68–86, 2012.

[8] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM J. Comput.*, 38(4):1602–1623, 2008.

[9] G. Avni, O. Kupferman, and T. Tamir. Network-formation games with regular objectives. In *Information and Computation*. 251, pages 165–178, 2016.

[10] G. Avni, O. Kupferman, and T. Tamir. Congestion games with multisets of resources and applications in synthesis. In *Proc. 35th FST&TCS*, LIPIcs, pages 365–379, 2015.

[11] T. Brihaye, V. Bruyère, J. De Pril, and H. Gimbert. On subgame perfection in quantitative reachability games. *LMCS*, 9(1), pages 1–32, 2012.

[12] K. Chatterjee. Nash equilibrium for upward-closed objectives. In *Proc. 15th CSL*, LNCS 4207, pages 271–286. Springer, 2006.

[13] K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Games with secure equilibria. *Theoretical Computer Science*, 365(1-2):67–82, 2006.

[14] K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. In *Proc. 18th CONCUR*, pages 59–73, 2007.

[15] K. Chatterjee, R. Majumdar, and M. Jurdzinski. On Nash equilibria in stochastic games. In *Proc. 13th CSL*, LNCS 3210, pages 26–40. Springer, 2004.

[16] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[17] H. Chen and T. Roughgarden. Network Design with Weighted Players, *Theory of Computing Systems*, 45(2), 302–324, 2009.

[18] W-P. de Roever, H. Langmaack, and A. Pnueli, editors. *Compositionality: The Significant Difference*, LNCS 1536, Springer, 1998.

[19] J. R. Correa, A. S. Schulz, and N. E. Stier-Moses. Selfish routing in capacitated networks. *Mathematics of Operations Research* 29: 961–976, 2004.

[20] D. Drusinsky and D. Harel. On the power of bounded concurrency I: Finite automata. *Journal of the ACM*, 41(3):517–539, 1994.

[21] A. Fabrikant, A. Luthra, E. Maneva, C. Papadimitriou, and S. Shenker. On a network creation game. In *ACM PODC*, pages 347–351, 2003.

[22] D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *Proc. 16th TACAS*, LNCS 6015, pages 190–204. Springer, 2010.

[23] M. Feldman and T. Tamir. Conflicting Congestion Effects in Resource Allocation Games. Journal of *Operations Research* 60(3), pages 529–540, 2012.

[24] D. S. Johnson, C. H. Papadimitriou and M. Yannakakis, How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[25] B. Kimelfeld and Y. Sagiv, New algorithms for computing Steiner trees for a fixed number of terminals, *http://www.cs.huji.ac.il/ bennyk/papers/steiner06.pdf*, 2006.

[26] E. Koutsoupias and C. Papadimitriou. Worst-case Equilibria. *Computer Science Review*, 3(2): 65-69, 2009.

[27] Y. Lustig and M.Y. Vardi. Synthesis from component libraries, STT&T, 15 (5-6): 603-618, 2013.

[28] I. Milchtaich. Weighted Congestion Games With Separable Preferences. *Games and Economic Behavior*, 67, 750-757, 2009.

[29] J. C. Mitchell. Concepts in programming languages, *Cambridge University Press*, 2003.

[30] M. Mavronicolas, I. Milchtaich, B. Monien, and K. Tiemann. Congestion Games with Player-specific Constants. In *Proc 32nd MFCS*, pp. 633–644, 2007.

[31] D. Monderer and L. Shapley. Potential Games. *Games and Economic Behavior*, 14:124–143, 1996.

[32] C. A. Meyers and A. S. Schulz. The Complexity of Welfare Maximization in Congestion Games. *Networks*, 59(2):252–260, 2012.

[33] C. H. Papadimitriou. Algorithms, games, and the internet. In *Proc. 33rd STOC*, pages 749–753, 2001.

[34] A. Pnueli. In transition from global to modular temporal reasoning about programs. In *Logics and Models of Concurrent Systems*, volume F-13 of *NATO Advanced Science Institutes*, pages 123–144. Springer, 1985.

[35] L. Rose, E.V. Belmega, W. Saad, and M. Debbah. Pricing in heterogeneous wireless networks: hierarchical games and dynamics. *IEEE Trans. Wireless Communications*, 13 (9): 4985–5001, 2014.

[36] W. Saad, Q. Zhu, T. Basar, Z. Han, and A. Hjørungnes. Hierarchical network formation games in the uplink of multi-hop wireless networks *Proc. GLOBECOM*, pages 1–6, IEEE, 2009.

[37] E. Tardos and T. Wexler. Chapter 19: Network formation games and the potential function method, In *Algorithmic Game Theory*, Cambridge University Press, 2007.