Hierarchical Network Formation Games

Orna Kupferman[1(✉)] and Tami Tamir[2]

[1] School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel
orna@cs.huji.ac.il

[2] School of Computer Science, The Interdisciplinary Center, Herzliya, Israel
tami@idc.ac.il

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 network design has become especially relevant with the emergence of

The research leading to this paper has received funding from the European Research Council (FP7/2007-2013)/ERC grant agreement No. 278410, and from The Israel Science Foundation (grant No. 1229/10). Due to lack of space, some proofs are omitted and can be found in the full version in the authors' URLs.

A. Legay and T. Margaria (Eds.): TACAS 2017, Part I, LNCS 10205, pp. 229–246, 2017.
DOI: 10.1007/978-3-662-54577-5_13

the Internet, where different users share resources like software or communication channels [1,7,18,21]. In *network-formation games* (NFGs, for short) [7], 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 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 reflected in the *price of stability* (PoS) [7], 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 [24,31,37].

To the best of our knowledge, all studies of NFGs consider an *explicit* representation of the network: it 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

[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.

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 [17]. 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 parallel and communicate with each other, possibly using variables [19,20,34]. 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. This, however, may involve an exponential blow up in the state space. In [5], 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 [6] 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 [4], 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 motivates 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

[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 [35,36], 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.

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 [27,30]. 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: 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 players' strategies 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 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 [7] in which players are weighted and a player's share in an edge cost is proportional to its weight is considered in [16], 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 [22,28,29,32]. 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 [2,12,13,23], an extension of NFGs to objectives that are richer than reachability [9], studies of non-zero-sum games in formal methods [11,15], augmentation of the problem of synthesis from component libraries with costs [8], 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 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.

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 $\pi = \{\langle s, b_1\rangle^1, \langle (b_1, u_4), b_2\rangle^1, \langle (b_2, u_4), t_2\rangle^1, \langle u_1, u_2\rangle^2, \langle u_2, u_4\rangle^2\}$.

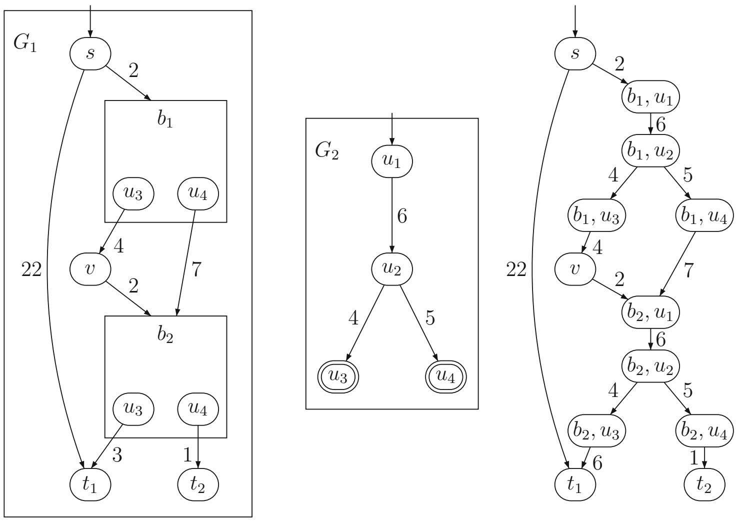

Fig. 1. An example of a hierarchical graph and its flat expansion.

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.

Observation 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.*

2.2 Network Formation Games

For an integer $k \in \mathbb{N}$, let $[k] = \{1, \ldots, k\}$. A *network-formation game* (NFG, for short) [7] 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 $1 \le i \le k$, 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 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 the number of players that use e in P by $load_P(e)$. 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 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) [7] 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, $\text{PoS}(\mathcal{N}) = \min_{P \in \Gamma(\mathcal{N})} cost(P)/OPT(\mathcal{N})$. 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, $\text{PoA}(\mathcal{N}) = \max_{P \in \Gamma(\mathcal{N})} cost(P)/OPT(\mathcal{N})$.

2.3 Hierarchical Network Formation Games

A *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 and study 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.

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 $\mathcal{G} = \langle G_1, G_2 \rangle$ in Fig. 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.

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.

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 denote 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 $wload_P(\langle v_1, b_1\rangle) = 2, wload_P(\langle u_1, u_2\rangle) = 4, wload_P(\langle u_2, u_3\rangle) = 3, wload_P(\langle (b_1, u_3), v\rangle) = 2, wload_P(\langle v, b_2\rangle) = 2$, 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 2. *Every HNFG with the flat or UH mechanism has an NE.*

For the PH mechanism, we present a negative result.

Theorem 3. *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 Fig. 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 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. □

A natural question arising from the above theorem is whether we can distinguish between instances that have or do not have a stable profile. In the full version of this paper we show that we can do it in Σ_P^2, yet this is an NP-hard task.

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.

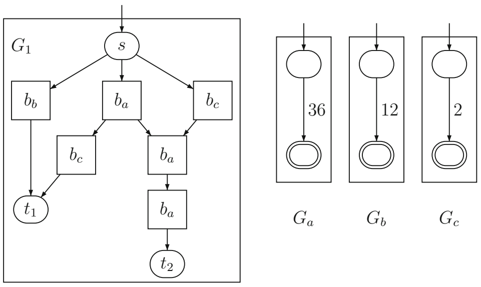

Fig. 2. An HNFG $\mathcal{N}$ that has no NE with the PH mechanism.

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

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

Theorem 4. *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 5. *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, whose cost is not larger than the cost of the whole SO profile. This contradicts 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.

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 Fig. 3, and the other components consists of $k/2$ copies of the graphs G_1, G_a, G_b, and G_c, described in Fig. 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 $1 \leq \ell \leq \frac{k}{2}$.

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.

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. 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. $\square$

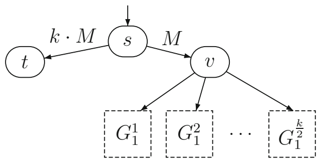

Fig. 3. An HNFG $\mathcal{N}_k$ for which PoS $= k$. Every G_1^ℓ is a copy of G_1 depicted in Fig. 2.

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 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 Sect. 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. Intuitively, in the UH mechanism, the proof of the dominance is easy, as by repeating the same path a player can only reduce the set of edges in his strategy, which results in reduced payment. Thus, in the UH mechanism, the used $\langle in_j, u\rangle$-path can be chosen arbitrarily. In the PH mechanism, the proof is more involved, as moving to the chosen $\langle in_j, u\rangle$-path may increase the payment for other uses of this path. Accordingly, not all choices of a $\langle in_j, u\rangle$-path are beneficial. We show, however, that at least one choice is beneficial. In addition, checking the feasibility of homogeneous strategies requires only one check for each subgraph G_j and exit vertex $u \in \mathit{Exit}_j$, which can be done in polynomial time. Hence we have the following:

Lemma 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 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 [31]. 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 6. *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.*

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 6 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.

Theorem 7. *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.*

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 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 [5,6]. 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 (as we proved in Theorem 7), 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 8. *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 9. *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 $j \in [n]$ 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 is omitted from this extended abstract. □

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 needs 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 10. *The problem of finding an NE in CE-HNFGs with the flat mechanism is PLS-complete.*

References

1. Albers, S., Elits, S., Even-Dar, E., Mansour, Y., Roditty, L.: On Nash equilibria for a network creation game. In: Proceedings of 7th SODA, pp. 89–98 (2006)
2. Almagor, S., Avni, G., Kupferman, O.: Repairing multi-player games. In: Proceedings of 26th CONCUR, pp. 325–339 (2015)
3. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002)

4. Alur, R., Kannan, S., Yannakakis, M.: Communicating hierarchical state machines. In: Wiedermann, J., Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 169–178. Springer, Heidelberg (1999). doi:10.1007/3-540-48523-6_14
5. Alur, R., Yannakakis, M.: Model checking of hierarchical state machines. ACM TOPLAS **23**(3), 273–303 (2001)
6. Aminof, B., Kupferman, O., Murano, A.: Improved model checking of hierarchical systems. J. Inf. Comput. **210**, 68–86 (2012)
7. 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. **38**(4), 1602–1623 (2008)
8. Avni, G., Kupferman, O.: Synthesis from component libraries with costs. In: Baldan, P., Gorla, D. (eds.) CONCUR 2014. LNCS, vol. 8704, pp. 156–172. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44584-6_12
9. Avni, G., Kupferman, O., Tamir, T.: Network-formation games with regular objectives. Inf. Comput. **251**, 165–178 (2016)
10. Avni, G., Kupferman, O., Tamir, T.: Congestion games with multisets of resources and applications in synthesis. In: Proceedings of 35th FST and TCS. LIPIcs, pp. 365–379 (2015)
11. Brihaye, T., Bruyère, V., De Pril, J., Gimbert, H.: On subgame perfection in quantitative reachability games. LMCS **9**(1), 1–32 (2012)
12. Chatterjee, K.: Nash equilibrium for upward-closed objectives. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 271–286. Springer, Heidelberg (2006). doi:10.1007/11874683_18
13. Chatterjee, K., Henzinger, T.A., Jurdzinski, M.: Games with secure equilibria. Theoret. Comput. Sci. **365**(1–2), 67–82 (2006)
14. Chatterjee, K., Henzinger, T.A., Piterman, N.: Strategy logic. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 59–73. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74407-8_5
15. Chatterjee, K., Majumdar, R., Jurdziński, M.: On Nash equilibria in stochastic games. In: Marcinkowski, J., Tarlecki, A. (eds.) CSL 2004. LNCS, vol. 3210, pp. 26–40. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30124-0_6
16. Chen, H., Roughgarden, T.: Network design with weighted players. Theor. Comput. Syst. **45**(2), 302–324 (2009)
17. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (1999)
18. Correa, J.R., Schulz, A.S., Stier-Moses, N.E.: Selfish routing in capacitated networks. Math. Oper. Res. **29**, 961–976 (2004)
19. de Roever, W.-P., Langmaack, H., Pnueli, A. (eds.): COMPOS 1997. LNCS, vol. 1536. Springer, Heidelberg (1998). doi:10.1007/3-540-49213-5
20. Drusinsky, D., Harel, D.: On the power of bounded concurrency I: finite automata. J. ACM **41**(3), 517–539 (1994)
21. Fabrikant, A., Luthra, A., Maneva, E., Papadimitriou, C., Shenker, S.: On a network creation game. In: ACM PODC, pp. 347–351 (2003)
22. Feldman, M., Tamir, T.: Conflicting congestion effects in resource allocation games. J. Oper. Res. **60**(3), 529–540 (2012)
23. Fisman, D., Kupferman, O., Lustig, Y.: Rational synthesis. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 190–204. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12002-2_16
24. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: How easy is local search? J. Comput. Syst. Sci. **37**, 79–100 (1988)

25. Kimelfeld, B., Sagiv, Y.: New algorithms for computing Steiner trees for a fixed number of terminals (2006). http://www.cs.huji.ac.il/bennyk/papers/steiner06.pdf
26. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. Comput. Sci. Rev. **3**(2), 65–69 (2009)
27. Lustig, Y., Vardi, M.Y.: Synthesis from component libraries. STTT **15**(5–6), 603–618 (2013)
28. Mavronicolas, M., Milchtaich, I., Monien, B., Tiemann, K.: Congestion games with player-specific constants. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 633–644. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74456-6_56
29. Milchtaich, I.: Weighted congestion games with separable preferences. Games Econ. Behav. **67**, 750–757 (2009)
30. Mitchell, J.C.: Concepts in Programming Languages. Cambridge University Press, Cambridge (2003)
31. Meyers, C.A., Schulz, A.S.: The complexity of welfare maximization in congestion games. Networks **59**(2), 252–260 (2012)
32. Monderer, D., Shapley, L.: Potential games. Games Econ. Behav. **14**, 124–143 (1996)
33. Papadimitriou, C.H.: Algorithms, games, and the internet. In: Proceedings of STOC, pp. 749–753 (2001)
34. Pnueli, A.: In transition from global to modular temporal reasoning about programs. In: Apt, K.R. (ed.) Logics and Models of Concurrent Systems. NATO Advanced Science Institutes, vol. 13, pp. 123–144. Springer, Heidelberg (1985). doi:10.1007/978-3-642-82453-1_5
35. Rose, L., Belmega, E.V., Saad, W., Debbah, M.: Pricing in heterogeneous wireless networks: hierarchical games and dynamics. IEEE Trans. Wireless Commun. **13**(9), 4985–5001 (2014)
36. Saad, W., Zhu, Q., Basar, T., Han, Z., Hjørungnes, A.: Hierarchical network formation games in the uplink of multi-hop wireless networks. In: Proceedings of GLOBECOM, pp. 1–6. IEEE (2009)
37. Tardos, E., Wexler, T.: Network formation games and the potential function method. In: Algorithmic Game Theory. Cambridge University Press (2007)