Timing Matters: Online Dynamics in Broadcast Games

SHUCHI CHAWLA, Computer Sciences Department, University of Wisconsin - Madison
JOSEPH (SEFFI) NAOR, Computer Science Department, Technion
DEBMALYA PANIGRAHI, Department of Computer Science, Duke University
MOHIT SINGH, H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology
SEEUN WILLIAM UMBOH, School of Computer Science, The University of Sydney

This article studies the equilibrium states that can be reached in a network design game via natural game dynamics. First, we show that an arbitrarily interleaved sequence of arrivals and departures of players can lead to a polynomially inefficient solution at equilibrium. This implies that the central controller must have some control over the timing of agent arrivals and departures to ensure efficiency of the system at equilibrium. Indeed, we give a complementary result showing that if the central controller is allowed to restore equilibrium after every set of arrivals/departures via improving moves, then the eventual equilibrium states reached have exponentially better efficiency.

CCS Concepts: • Theory of computation → Approximation algorithms analysis; Online algorithms; Quality of equilibria; Network games; Network formation;

Additional Key Words and Phrases: Broadcast games, game dynamics, online algorithms

ACM Reference format:
Shuchi Chawla, Joseph (Seffi) Naor, Debmalya Panigrahi, Mohit Singh, and Seeun William Umboh. 2021. Timing Matters: Online Dynamics in Broadcast Games. ACM Trans. Econ. Comput. 9, 2, Article 11 (May 2021), 22 pages. https://doi.org/10.1145/3434425

1 INTRODUCTION

In multi-agent systems where different agents have competing objectives, it is well-known that selfish behavior leads to suboptimal system performance at equilibrium. The Price of Anarchy

Partial support for this work was provided by the following grants: S. Chawla from NSF grants CCF-1101429 and CCF-1320854; S. Naor from ISF grant 1585/15 and BSF grant 2014414; D. Panigrahi from NSF grants CCF-1527084 and CCF-1535972, an NSF CAREER Award CCF-1750140, and faculty research awards from Google and Yahoo; M. Singh from NSF grant CCF-1717947; S. Umboh from ERC consolidator grant 617951 and NSF grant CCF-1320854.

Authors’ addresses: S. Chawla, Computer Sciences Department, University of Wisconsin - Madison, 1210 W. Dayton Street, Madison, WI 53706, USA; email: shuchi@cs.wisc.edu; J. (Seffi) Naor, Computer Science Department, CS Taub Building, Technion Israel, Institute of Technology, Haifa 3200003, Israel; email: naor@cs.technion.ac.il; D. Panigrahi, Duke University, Campus Box 90129, 308 Research Drive (LSRC Building), Room D203, Durham, NC 27708, USA; email: debmalya@cs.duke.edu; M. Singh, H. Milton Stewart School of Industrial and Systems Engineering, 755 Ferst Drive, NW, Atlanta, GA, 30332-0205, USA; email: mohitsingh@gmail.com; S. W. Umboh, School of Computer Science, The University of Sydney, Building J12/1 Cleveland St, Camperdown NSW 2006 Australia; email: william.umboh@sydney.edu.au.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
2167-8375/2021/05-ART11 $15.00
https://doi.org/10.1145/3434425
(PoA) and the Price of Stability (PoS), which, respectively, correspond to the worst and best equilibrium states, are widely used in the literature to quantify this suboptimality relative to an optimal solution designed by a central authority. If these two measures are close to each other, then they provide a satisfactory understanding of the quality of stable states the system is expected to reach. However, when these measures differ significantly, the system can exhibit multiple equilibria with highly varying performance. But, which of these equilibria can be achieved in actual game dynamics? More generally, what is the minimal guidance by a central authority that can guarantee near-optimal system performance in equilibrium?

In this article, we study these questions in the context of a game that exhibits a particularly rich set of equilibria, namely, the broadcast game. A broadcast game is defined on a rooted undirected graph with costs on edges. Every vertex in the graph has an agent whose goal is to select a routing path to the root that minimizes her own cost. The cost of every edge is shared equally among all agents using it, and the cost of an agent is the sum of her cost shares along the edges in her path to the root. The system is in Nash equilibrium (NE) if no agent can lower her own cost by unilaterally changing her routing path. The cost of an equilibrium is the total cost of all edges used by at least one agent. The quality of equilibria is measured with respect to the social optimum, which for broadcast games is the minimum spanning tree (mst) of the graph.

Broadcast games are a kind of potential games and the existence of NE in any instance can be proved through a potential function argument, originally given by Rosenthal [33] (see also Monderer and Shapley [29]). Anshelevich et al. [4] observed that different NE in broadcast games can exhibit vastly different performance: The PoA can be as large as $\Omega(n)$, whereas the PoS (a concept they introduced to show this gap) is bounded by $O(\log n)$; here, $n$ denotes the number of vertices in the graph.\(^1\) A long line of works [10, 18, 26, 27] subsequently improved the PoS bound to $O(1)$.

Given this divergence of bounds, Chekur et al. [12] posed the question of analyzing the quality of equilibria that are actually reachable via natural dynamics—a sequence of single agent moves where the moving agent always chooses a new path that strictly decreases her cost relative to her current path. We call such moves “improving moves” or “best response moves,” depending on whether they merely lower the agent’s cost or are optimal for the agent given the current state of the system. It follows from the potential function argument of Rosenthal [33] that any such sequence of moves will eventually converge to NE.

Chekur et al. [12] considered the following restricted two-stage process: In the first stage, starting with an empty graph, agents arrive sequentially in arbitrary order and choose their respective best response paths upon arrival; in the second stage, agents make improving moves\(^2\) in arbitrary order until they reach equilibrium. They showed that the equilibria reachable through this process have a cost of $O(\sqrt{n\log^2 n})$ times the mst, a significant improvement over the PoA bound. This bound was subsequently improved to $O(\log^3 n)$ for the same two-stage process by Charikar et al. [11].

The dynamic price of stability. These previous works motivate extending the notion of the price of stability to online dynamics. In the static (or “one shot”) version of our problem, in which players are initially in an empty configuration, the central planner can force the players into any configuration, in particular the one realizing the price of stability. In the dynamic case, however, the central planner cannot do so, since some players have already chosen a route. Thus, the central planner has to offer existing players a better strategy to incentivize changes. Informally, the dynamic price of stability is the cost of a solution in equilibrium resulting from online dynamics, while allowing

\(^1\)See Appendix B for examples illustrating these bounds.
\(^2\)Observe that when an agent arrives or makes an improving move, paths of other agents may become suboptimal for them.
for algorithmic intervention by the central planner. The notion of dynamic price of stability can be applied to any game in which one needs to characterize which equilibria can be reached via online dynamics, while minimizing the power of intervention of the central planner. It would be very interesting to find further applications of this new notion.

One way to restate Charikar et al.’s result [11] is that the dynamic PoS is polylogarithmic when all arrivals happen before any improving moves. But, what if some agents make improving moves before all of the other agents have arrived, i.e., the sequence of improving moves is interleaved with arrivals? Unfortunately, the analyses presented in References [12] and [11] strongly build on the fact that all agents arrive upfront and remain in the system thereafter, and thus agents must wait for everyone to arrive before making any changes to their strategies. Charikar et al. posed the question of analyzing dynamics in which arrivals and improving moves are interleaved as a “tantalizing and difficult” open problem. In the decade following their work, in spite of tremendous progress in PoS bounds for broadcast games, no progress has been made on understanding more general game dynamics.3

More general dynamics. Since the work of Charikar et al. [11], our work is the first to study more general dynamics of the broadcast game. We consider two kinds of extensions to the two-stage process. First, we consider systems with churn where agents arrive as well as depart over time. Second, we allow multiple interleaved stages of arrivals, departures, and improving moves. Our first result shows that if we make a minimal change to the two-stage dynamics studied above, namely, adding departures to the first stage, then it is possible to reach an equilibrium that is polynomial (in n) worse than the social optimum, placing it in the same regime as the PoA bound. To the best of our knowledge, this is the first polynomial lower bound for any game dynamics for broadcast games.

**Theorem 1.1.** For any large enough integer n, there exists an instance of the broadcast game with n vertices and a sequence of arrivals and departures that terminates in a NE of cost $\Omega(n^{1/3})$ times that of the minimum spanning tree on all the vertices.

It is important to observe that, since we allow departures, not all vertices have agents at the end of the game. This creates two candidates for $\text{opt}$: the optimal Steiner tree on the remaining agents, or the MST on all vertices.4 The former leads to trivial and uninteresting lower bounds (see Appendix B); so, we use the MST as opt in this article. This choice of a weaker optimum makes for a stronger lower bound.

The power of intervention. Given the above lower bound, a natural question is whether some limited intervention from a central planner can lead to a better outcome for the game. At one extreme, if the central planner is allowed to suggest a strategy to every player simultaneously, then any NE, in particular the best one corresponding to the PoS of O(1), can be achieved. This level of control is unrealistic. A more reasonable level of control is for the central planner to suggest improving moves to players one-by-one; importantly, any such move should lower the corresponding agent’s current cost share, otherwise the player has no incentive to follow the planner’s suggestion.

What about the timing of such interventions? As our lower bound demonstrates, if the timing of interventions is completely adversarial, in particular if no interventions are allowed during the

---

3Charikar et al. [11] also studied a variant where arrivals happen in uniformly random order and are interleaved with adversarially ordered best response moves. For this setting, they were able to prove an upper bound of $O(\sqrt{n} \text{polylog } n)$ on the quality of the equilibria reached, but did not present any lower bounds.

4Another bound is the optimal Steiner tree on all vertices for which an agent arrived at some point in the dynamics. Since we can assume metric costs, we can restrict our attention to these vertices and then the MST cost is within a factor of two of the cost of an optimal Steiner tree.
initial arrival/departure phase, then the system can end up in a poor NE. To get around this lower bound, we consider dynamics where the central planner is allowed to make a series of improving moves after every adversarial arrival/departure. Observe that the sequence of arrivals and departures can still be ordered adversarially, and indeed can depend on the previous algorithmic interventions. We call such dynamics \textit{equilibrium-preserving} (EQ-P) dynamics, because the central planner restores the system to a good equilibrium after every adversarial arrival/departure.

Specifically, the EQ-P dynamics starts from an empty configuration and continues in epochs. At the beginning of each epoch the system is at equilibrium. The epoch begins with an arrival or departure, followed by a series of improving moves to restore equilibrium. Once equilibrium is restored, the epoch ends. Our analysis, in fact, allows for multiple simultaneous arrivals\(^5\) at the beginning of an epoch, and multiple departures at any point of time during the epoch. Formally, we define three different kinds of moves within an epoch:

(1) \textbf{(Arrivals.)} A set of new players arrives and each player picks a best response path with respect to the configuration reached at the end of the previous epoch. (The choice of the set of arrivals is adversarial.)

(2) \textbf{(Departures.)} A set of players departs the system. (The choice of departing players is adversarial.)

(3) \textbf{(Equilibrium Restoration.)} The central authority offers players strategies that can improve their (shared) connection costs. This step continues until equilibrium is restored to the system.

Our second result shows that this limited level of central intervention is sufficient to guarantee an NE with exponentially better performance:

**Theorem 1.2.** Every instance of the broadcast game using EQ-P dynamics converges to a NE of cost $O(\log n)$ times that of the minimum spanning tree on all the vertices.

Observe that, as for our lower bound result, we compare the performance of EQ-P dynamics to the MST on all vertices\(^6\) and not the optimal Steiner tree on the vertices that remain in the system. The two benchmarks are identical when there are no departures, but the MST benchmark can potentially be much weaker when there are many departures. However, as mentioned earlier, the Steiner tree benchmark is not interesting, because it leads to trivial polynomial lower bounds (see Appendix B). Furthermore, the guarantee provided by the above theorem holds at the end of every epoch as compared against the MST over vertices that have arrived up to the end of that epoch, not including future arrivals. A natural open question is whether a polylogarithmic dynamic PoS can be achieved through less algorithmic intervention relative to EQ-P dynamics; for example, by allowing players to make best response moves instead of improving moves.

**Related Work.** We have already mentioned the long line of work on improving PoS bounds for broadcast games [4, 10, 18, 26], and the game dynamics studied by Chekuri et al. [12] and Charikar et al. [11]. A different approach was taken by Balcan et al. [6], who considered the problem of influencing the dynamics of broadcast games to achieve socially efficient equilibria. In their model, players use expert learning, choosing between a best response expert and a central authority expert suggesting (near-)optimal global behavior. Broadcast games belong to a broader class called

\(^5\)Note that although arrivals within a single epoch are simultaneous in that every arriving player picks a best response path with respect to the equilibrium state at the beginning of the epoch, arrivals in different epochs are sequential. In this sense, our model captures sequential arrivals with interleaved improving moves.

\(^6\)Because of this comparison against the MST, we prefer the term “broadcast game” for this setting, rather than “multicast game.”
network design games (see, e.g., References [2, 4, 9, 10, 13, 14, 17, 19, 25, 27]), which in turn, are a special case of the widely studied congestion and potential games (see, e.g., References [1, 7, 16, 23, 28, 29, 32, 33, 35]).

The analysis of game dynamics in this article crucially relies on the construction of a hierarchical family of multiple dual solutions. This method of analysis has been highly influential in designing online algorithms for network design problems. Implicit use of this method dates back to the work of Imase and Waxman [24] on online Steiner trees and a subsequent line of work of Awerbuch et al. [5], Berman and Coulston [8], Naor et al. [30]. More recently, this method has been explicitly employed in solving a range of node and edge-weighted Steiner network design problems in the online setting [3, 15, 20–22, 31]. In terms of the exact techniques, perhaps the closest to our work is that of Umboh [36], who uses hierarchical tree embeddings to analyze greedy-like online algorithms for network design problems. In contrast to these applications in competitive analysis, where decisions are irrevocable, game dynamics allows temporary overcharging of dual solutions, which we crucially use in this article.

An interesting related problem to our line of work is determining the computational status of finding a NE solution of broadcast games. To the best of our knowledge this problem is open and neither hardness results, nor algorithmic upper bounds, are known. We mention several related results. It was shown by Chekuri et al. [12] that it is NP-hard to compute the global potential minimizer for multicast games (in which not all vertices necessarily contain an agent). Syrgkanis [34] showed that it is PLS-complete to compute a NE solution in network cost-sharing games. In these games there is no common root, but rather a distinct destination for each agent. Thus, these games are more general than our broadcast game. We note that the PLS-hardness proof seems to strongly use the fact that in the network game there are many destinations.

Organization of the article. We begin by defining a model for the broadcast game, as well as a charging scheme to analyze the cost of equilibria, in Section 2. We present a proof of our lower bound (Theorem 1.1) in Section 3 and a proof of our upper bound (Theorem 1.2) in Sections 4 and 5.

2 MODEL AND TECHNICAL BASICS

The broadcast game is defined on a complete graph $G = (V, E)$, $|V| = n$, with metric costs $c : V \times V \to \mathbb{R}_+$ defined on the edges and a root vertex $r$. We assume without loss of generality that every vertex has a unique agent, or terminal, residing at it. Agents arrive and depart over time. Since edge costs satisfy the triangle inequality, before an agent arrives, no other agents route their paths via the vertex corresponding to this agent. However, if an agent who is already in the system departs, then other agents may continue to route their paths via its vertex, and the vertex remains in the graph. We call a vertex active if the agent at that vertex is still in the system.

Let $t$ index time. We use $V_t$ to denote the set of all vertices in $G$ that have appeared up to time $t$. Let $A_t \subseteq V_t$ denote the set of active terminals among them. Each terminal $v \in A_t$ has a current routing path $p_v$ connecting it to the common root $r$. The cost share of $v$ along this path is the sum of $v$’s cost share over the edges in the path, where the cost of an edge is equally shared between all terminals currently using it.

The routing at any time $t$ is defined to be the set of routing paths $(p_v)_{v \in A_t}$. A best response path of a terminal $v$ with respect to a routing, denoted $p_v^*$, is a path from $v$ to $r$ with the minimum shared cost if $v$ were to move to this path. If there are multiple such paths, then we break ties in favor of
paths having fewer edges with no terminal other than \( v \) using them. Note that this may not break all ties, in which case, any of these paths can be designated as the best response path. A terminal \( v \in A \) is said to have an improving move with respect to a routing if by moving from its current path \( p_v \) to a new path \( q_v \) strictly decreases \( v \)'s cost share. Given a routing, its potential [33] is defined to be \( \Phi = \sum_{e \in E} \sum_{i=1}^{N_e} c_e / i \), where \( N_e \) is the number of agents using \( e \). A standard argument shows that any improving move decreases the potential by a value that is uniformly bounded away from zero, resulting in a finite convergence of any improving move-based dynamics.

The following well-known lemma states that in equilibrium, the routing paths form a tree:

**Lemma 2.1.** In equilibrium, the routing paths of a broadcast game form a tree.

**Proof.** The lemma is a direct consequence of the following downward closure property that holds in an equilibrium state. Suppose \( w \) is a vertex (not necessarily a terminal) that appears on the routing paths \( p_u \) and \( p_v \) of two terminals \( u \) and \( v \), respectively. Then, the segment of \( p_u \) and \( p_v \) between \( w \) and \( r \) must be identical. For the sake of contradiction, suppose this claim is false, and let \( q_u \) and \( q_v \) denote the non-identical segments of \( p_u \) and \( p_v \) between \( w \) and \( r \). Assume without loss of generality that the current shared cost of \( q_u \) is at most that of \( q_v \). If \( v \) now moves to a new routing path that follows \( p_u \) until \( w \) and then uses \( q_u \) to reach \( r \), then the change in cost of \( v \) will be the difference of the new shared cost of \( q_u \) and the current shared cost of \( q_v \). This is clearly non-positive by our assumption on the relative order of current shared costs of \( q_u \) and \( q_v \). In fact, we argue that this difference is negative. First, note that there is at least one edge that is in \( q_u \), but not in \( q_v \), since the two paths are non-identical. Now, let \((x, y)\) be the closest edge to \( u \) in \( q_u \) that does not appear in \( q_v \), where \( x \) is closer to \( u \) than \( y \). Then, \( x \) must also appear on \( q_v \), and therefore, cannot appear in the segment of \( p_v \) between \( v \) and \( w \). It follows that edge \((x, y)\) does not appear in the segment between \( v \) and \( w \) in \( p_v \), and hence is absent from the entire path \( p_v \). When \( v \) moves to its new path, the shared cost on \((x, y)\) decreases below its current value, and therefore, the shared cost of \( q_u \) decreases as a whole. This implies that \( v \) has an improving move, which contradicts the premise that the terminals are in equilibrium.

\(\square\)

### 2.1 A Charging Scheme for Tree Routings

Our goal is to compare the cost of equilibria arising in the broadcast game to the cost of the MST of the graph. At any point of time \( t \) when equilibrium is reached, we will compare the cost of the current routing to the MST over the set \( V_t \) of vertices. To do so, we consider the standard MST linear program and its dual. We now describe our charging scheme in detail. The dual defines a packing of vertex cuts into edges. It is not necessary to understand the MST program or its dual. We describe it directly in the form of cut packings.

We call a partition \( P = (S_1, \ldots, S_m) \) of the vertex set \( V \) a level-\( j \) dual for an integer \( j \) if it satisfies the following:

- \( P \) is a partition: \( \cup_{S \in P} S = V \), and for any \( S_a, S_b \in P \), \( S_a \cap S_b = \emptyset \).
- The components have bounded diameter: For any \( S \in P \), and any vertices \( x, y \in S \), \( c(x, y) < 2^j \).
- The components are far from each other: There exists a “center” vertex \( s_i \) in each component \( S_i \), such that for all \( S_a, S_b \in P \), \( c(s_a, s_b) \geq 2^{j-1} \).

We use the term \( \text{cuts} \) to denote the components \((S_1, \ldots, S_m)\) of the partition. The lemma below follows immediately from the observation that any spanning tree over \( V \) must connect the centers of all cuts in a level-\( j \) dual \( P \).

**Lemma 2.2.** For any level-\( j \) dual \( P \), the cost of the minimum spanning tree \( \text{OPT} \) is at least \( 2^{j-1}(|P| - 1) \).
To bound the cost of any tree-routing equilibrium, we relate the cost of the edges used in the solution to a family of duals. Let $\Pi = \{P_j\}_{j \in \mathbb{Z}}$ denote a family of partitions, where $P_j$ is a level-$j$ dual.

Our charging scheme for routing solutions that form a tree proceeds as follows: Every vertex in the routing tree is responsible for the cost of its parent edge. Consider an edge $e = (v, \text{parent}(v))$ with length in $[2^{j+2}, 2^{j+3})$ for some $j \in \mathbb{Z}$. We charge the cost of this edge to the cut in the level-$j$ dual that contains $v$: i.e., $S \in P_j$ such that $v \in S$. Our final bound on the cost of the solution will now depend on how many times each cut in the family $\Pi$ gets charged by the edges in the tree, as formalized in the lemma below.

For non-EQ-P dynamics, we will construct an example with an equilibrium tree routing where at every scale we can find many vertices in close proximity that have parent edges of similar long length. This implies that any family of duals will get charged many times, leading to a large overall cost. For EQ-P dynamics, however, we will show that the solutions that result from algorithmic interventions will always charge each cut in a particular family $\Pi$ at most once.

**Lemma 2.3.** Suppose that our charging scheme charges each cut in the family $\Pi$ at most $\alpha$ times. Then the cost of the solution is at most $O(\alpha \log n \text{opt})$.

**Proof.** Let $D$ denote the largest edge length in the graph. Then, we first note that we may ignore edges in our solution of length at most $D/n$. This is because there are at most $n$ such edges, and opt is at least $D$ by the metric property of edge lengths. For the remainder, we charge duals at levels $j$ for $j \in (\log(D/n) - 3, \log D - 2]$. There are at most $\log n + 1$ such duals. The total cost of edges charged to a dual at level-$j$ is at most $2^{j+3}|P| < 322^{j-1}(|P| - 1)$. Lemma 2.2 then implies the result. □

For much of our analysis, we assume that the dual family $\Pi$ is provided to us. In Appendix A, we discuss how to construct this family algorithmically as terminals arrive online.

### 2.2 Technical Challenges in Analyzing EQ-P Dynamics

The broadcast game exhibits a rich set of equilibria and a far richer set of intermediate states of the system. Whereas the routing at any equilibrium of the game always forms a tree, intermediate states, even those reached by a series of best response moves, can contain a complex structure of interconnected cycles. A major impediment to analyzing dynamics is that it is extremely challenging to maintain any structural invariant on intermediate states. Our work overcomes this challenge by algorithmically maintaining such a structural invariant. Whenever the structural invariant is broken by arrivals or departures, we restore it algorithmically. Importantly, we show that this can always be achieved through a sequence of improving moves.

Our structural invariant is a charging of the cost of a state (i.e., collection of paths) against a family of partitions $\Pi$ of the underlying graph as defined above. As such, our charging scheme can be interpreted as a dual fitting approach. A feasible charging is one where each cut in the partition $\Pi$ is charged at most once. One challenge in carrying out this approach is that as agents arrive and leave, our analysis must allow for the charging to become grossly infeasible at intermediate states, which in turn results in very expensive intermediate (non-equilibrium) states. At the crux of our argument is a careful construction of improving moves that ensures that the system cycles between a small set of states of which the stable ones correspond to feasible chargings.

### 3 LOWER BOUND FOR non-EQ-P DYNAMICS

In this section, we will show that if arrivals and departures are allowed at non-equilibrium states, then no dynamics can lead to a good equilibrium (Theorem 1.1).
We construct a family of lower bound instances parameterized by an integer $m \geq 1$. The $m$th instance uses the metric induced by weighted graph $G_m$ (see Figure 1). The vertex set of this graph consists of a root $r$ and $m+1$ layers $V^0, \ldots, V^m$. For $1 \leq i \leq m$, layer $V^i$ consists of $m$ clusters $C^i_1, \ldots, C^i_m$, each of which is a clique over $m$ vertices. We use $v^i_{j,k}$ to denote the $k$th vertex of $C^i_j$; recall that each of $i, j,$ and $k$ take on integral values in $[m]$. Layer $V^0$ also consists of $m^2$ vertices, which are labeled $v^0_{j,k}$ for $j, k \in [m]$, but there are no edges between these vertices. The vertices of $V^m$ are called end vertices, and those of $V^0$ are called auxiliary vertices. Observe that the graph $G_m$ has $n = m^2(m+1) + 1$ vertices in all.

Next, we describe the edges. Each pair of vertices within the same cluster $C^i_j$ is connected by an edge of length $1/m$ for all layers except $V^0$. The remaining edges in the graph connect vertices in neighboring layers and are all of length 1. Each auxiliary vertex $v^0_{j,k}$ in $V_0$ is connected to the root and to its corresponding vertex $v^1_{j,k}$ in layer 1. For $1 \leq i \leq m-1$, we have an edge $(v^i_{j,k}, v^{i+1}_{k,j})$ for each $j, k \in [m]$. In other words, the vertices of the $j$th cluster in layer $i$ are connected to the $k$th vertices of the clusters in layer $i+1$; in particular, the $k$th vertex of the $j$th cluster in layer $i$ is connected to the $j$th vertex of the $k$th cluster in layer $i+1$. For example, see the edges leaving the first (top) cluster of $V_1$ in Figure 1. Observe that there are exactly $m^2(m+1)$ inter-layer edges, and exactly $m^3(m-1)/2$ intra-cluster edges.

Observe that each end vertex $v^m_{j,k}$ has a unique path $P_{j,k}$ to the root that consists of only inter-layer edges (see Figure 1). We call these paths canonical paths. Note that each inter-layer edge belongs to exactly one canonical path. In other words, the set of inter-layer edges is a disjoint union of all the canonical paths $P_{j,k}$.

The cost of the final equilibrium. Before describing the sequence of arrivals and departures of terminals, let us analyze the final equilibrium state and its cost relative to the optimal cost. Let OPT denote the cost of the minimum spanning tree over all vertices in $G_m$. Observe that this is an
upper bound on the cost of any optimal solution at the end. The final state following our sequence of arrivals and departures, denoted $F$, consists of $m$ players situated at every end vertex $v_{j,k}$ in layer $m$; each player uses the canonical path $P_{j,k}$ to route to the root. The following lemma shows that this is an equilibrium state with a polynomially larger cost relative to OPT.

**Lemma 3.1.** State $F$ is an equilibrium and the cost of $F$ is $\Omega(m)$ OPT.

**Proof.** First, we prove that $F$ is an equilibrium. Consider a player at end vertex $v_{j,k}^m$ with path $P_{j,k}$ and an alternative path $p'$. For every intra-cluster edge $e$, we have $N_e(F) = 0$, and for every inter-layer edge $e$, we have $N_e(F) = m$, where $N_e$ denotes the number of terminals using edge $e$. So the player’s current cost share is $m+1$. Path $p'$ contains at least one intra-cluster edge and at least $m + 1$ inter-layer edges. Thus, the player’s cost share when it switches to $p'$ is at least $\frac{1}{m} + \frac{m+1}{m+1} = \frac{m+1}{m}$. Therefore, $F$ is an equilibrium state.

Next, we prove that $c(F) = \Omega(m)$ OPT. $F$ consists of all unit-length inter-layer edges, and therefore its cost is $m^2(m + 1)$. However, one way of constructing a spanning tree for $G_m$ is to select an arbitrary spanning tree within each of the $m^2$ cliques, all of the edges from layer 0 to the root, as well as one inter-layer edge per clique connecting it to its preceding layer, say, the edge $(v_{j,k}^1, v_{j,k}^{i+1})$ to connect $C_{j+1}^i$ to $C_j^i$ for all $i, j \in [m]$. The total cost of this solution is $\frac{1}{m}m^2 + m^2 + m^2 = 3m^2$. □

**Sequence of arrivals and departures.** The sequence is constructed in $m$ phases, each phase consisting of $m^2$ rounds, one per end vertex $v_{j,k}^m$, and indexed by $(j, k)$. Informally, the objective of each round is to add one more terminal at each of the end vertices $v_{j,k}^m$. At the end of all the phases, there will be $m^3$ terminals. Within round $(j, k)$ in a phase, we use a set of “temporary” terminals whose sole aim is to force the terminal at $v_{j,k}^m$ that arrives at the end of the round to choose the canonical path as its best response. The temporary terminals are introduced at intermediate vertices along the canonical path during the round and removed at the end of the round.

Formally, let $< \cdot$ be an arbitrary total order on the pairs $(j, k)$. The sequence $\sigma$ is constructed to maintain the following invariant: At the end of round $(j, k)$ of phase $\ell$, there will be $\ell$ players on $v_{j,k}^m$, for $(j', k') < (j, k)$, and $\ell - 1$ players on the remaining end vertices. Furthermore, each player on $v_{j,k}^m$ uses the path $P_{j,k}$.

We now specify the subsequence for each round. Consider round $(j, k)$ of phase $\ell$. For simplicity of notation, we use $v^i$ to denote the vertex of $V^i$ on $P_{j,k}$. We also use $P^i$ to denote the segment of $P_{j,k}$ starting at $v^i$ and ending at the root. The round consists of $m + 1$ iterations. In iteration $0 \leq i \leq m - 1$, $m^2$ players arrive at $v^i$. In iteration $i = m$, one player arrives at $v^m$. Finally, the players on $v^0, \ldots, v^{m-1}$ depart. We can now show, using induction over the terminal arrivals, that for every terminal the best-response path on arrival is the segment of the canonical path connecting it to the root.

**Lemma 3.2.** Consider a terminal arriving at vertex $v^i$ in iteration $i$ of round $(j, k)$ in phase $\ell$. The best-response path of the terminal to the root is the segment of its canonical path $P^i$.

**Proof.** We prove the invariant by induction over the sequence of terminal arrivals, i.e., over $(\ell, j, k, i)$. Moreover, we will only prove that the inductive hypothesis holds for the first terminal arriving in an iteration; every subsequent terminal will clearly choose the same path as the first terminal, since they are arriving at the same vertex. The invariant trivially holds prior to the start of round 1 of phase 1.

Now consider the start of iteration $i$ of round $(j, k)$ in phase $\ell$ and assume the invariant held in previous iterations, rounds, and phases. Let us count the number of terminals on each edge. So far, on each end vertex, there are either $\ell$ or $\ell - 1$ terminals and each of them chose their canonical path.
on arrival, by the inductive hypothesis. The inductive hypothesis also tells us that for each \( i' < i \), there are at least \( m^2 \) terminals on \( v' \) using the path \( P^{i'} \). Thus, each inter-layer edge belonging to \( P^{i-1} \) has at least \( m^2 \) terminals, and each inter-layer edge that does not belong to \( P^{i-1} \) has at most \( \ell \) terminals. Moreover, none of the intra-cluster edges are used by any terminal.

Note that \( P^i \) consists of the inter-layer edge \((v^i, v^{i-1})\) followed by \( P^{i-1} \). Since \( |P^{i-1}| \leq m \), the cost share of the new terminal at \( v^i \) (call this terminal \( a \)) on \( P^i \) is at most \( 1/\ell + m/(m^2 + 1) < 1/\ell + 1/m \). Any other path \( Q \) for \( a \) contains at least two inter-layer edges that do not belong to \( P^{i-1} \) and at least one intra-cluster edge, so \( a \)'s cost share on \( Q \) is at least \( 2/(\ell + 1) + 1/m \geq 1/\ell + 1/m \). Thus, player \( a \)'s unique best-response path is \( P^i \).

Lemma 3.2 shows that the sequence of arrivals and departures above terminates in the final state \( F \), which costs \( \Omega(m) \) OPT by Lemma 3.1. Since \( m \) is polynomial in the number of vertices, Theorem 1.1 follows.

### 4 EQ-P DYNAMICS

In this section and the next, we describe and analyze EQ-P dynamics for the broadcast game. We first prove some basic structural properties that are used in the rest of the article.

In EQ-P dynamics, graph \( G \) is revealed via an online process that is divided into epochs (indexed by time \( t \)). At the start of epoch \( t \), the set of all vertices that have appeared so far and the set of active vertices are denoted \( V_t \) and \( A_t \), respectively. At the start of the epoch, we further enforce the invariant that the set of paths \( p_u \) are in NE, i.e., no terminal has an incentive to unilaterally deviate to a different routing path. We assume that our intervention algorithm has no knowledge of vertices corresponding to agents that are yet to arrive. At any point of time during the process, our algorithm only considers the subgraph induced over vertices whose agents have arrived prior to that time.

### 4.1 Routing Trees and Improving Moves

Each epoch \( t \) is divided into several phases. The first phase consists of an arrival or departure event. In the former case, a new set of terminals \( U_t \subseteq V \setminus V_t \) arrive, and the cost of all edges incident on terminals in \( U_t \) is revealed. Each new terminal \( u \in U_t \) chooses a best response routing path \( p_u \). In the latter case, a set of terminals leave, thereby removing the corresponding vertices from the set of terminals \( A_t \). (Note that the corresponding vertices remain in \( V_t \).) Lemma 5.2 establishes that the structure of the set of routing paths after arrivals or departures remains a tree.

Both arrival and departure events lead to changes in the cost shares of edges. In the EQ-P scenario, this might lead to a violation of the equilibrium state that was being previously maintained. In this case, the system performs a sequence of improving moves, in each of which a terminal changes its routing path to reduce its cost share.

Improving moves may temporarily create cycles in the collection of routing paths \( \{p_v\}_{v \in A_t} \). We order and group improving moves into contiguous blocks or phases such that every phase ends with the routing paths forming a tree. Furthermore, the trees at the beginning and end of the phase differ in a single pair of edges. The collection of moves in each such phase is called a tree-follow move.

**Definition 4.1 (Tree-follow Move).** A tree-follow move from \( u \) to \( v \) in \( T \) is a sequence of improving moves that start with routing tree \( T \) and end with routing tree \( T' = T \setminus \{u, \text{parent}(u)\} \cup \{u, v\} \), where parent\((u)\) is the parent vertex of \( u \) in \( T \). Observe that each terminal in the subtree rooted at \( u \) in \( T \) reroutes its path to the root according to \( T' \). (See Figure 2 below.)

Because of departure events, the routing tree may contain non-terminal vertices as Steiner vertices. It is convenient to extend the notion of an improving move to vertices that are not terminals.
Timing Matters: Online Dynamics in Broadcast Games

Fig. 2. The left panel shows a tree-follow move from $u$ to $v$. The right panel illustrates the individual improving moves that constitute this tree-follow move.

Let $w \not\in A$ be a non-terminal vertex. We say that $w$ has an improving move if the following properties hold: (1) There exists a terminal $v$ whose routing path $p_v$ includes $w$; let $p_w$ denote the segment of $p_v$ between $w$ and $r$; (2) There exists a path $q_w$ between $w$ and $r$ such that if $v$ were to retain its current routing path from $v$ to $w$ but move from $p_w$ to $q_w$, then the cost share of $v$ would strictly decrease.

A priori, it is not clear whether improving moves can always be grouped into tree-follow moves. In Lemma 5.3, we show that in every routing tree $T$ that is not in equilibrium there exists a sequence of improving moves that collectively form the tree-follow move from $u$ to $v$ for some vertices $u$ and $v$. When there are multiple such moves, we use a careful charging scheme to identify the order in which tree-follow moves should be implemented. (See Algorithm select tree move defined at the end of this section.)

Since every vertex in a tree has a unique path to the root, it suffices to specify the tree itself in lieu of all of the routing paths. Henceforth, we will use $T_t$ to denote the tree induced by $\{p_v\}_{v \in A_t}$ without explicitly specifying the paths themselves.

**EQ-P Dynamics**

1. Initialization. $t = 1$, $V_0 = \{r\}$, $T_0 = \{r\}$, $A_0 = \emptyset$.
2. For $t = 1, 2, \ldots$
   - (Arrivals.) Let $U_t$ be the set of terminals arriving. Let $A_t \leftarrow A_{t-1} \cup U_t$. For each $v \in U_t$, let $p_v = p_v^*$ where $p_v^*$ is the best response path of $v$ with respect to $T_{t-1}$. Let $T_t = T_{t-1} \cup_{v \in U_t} p_v^*$.
   - (Departures.) Let $D_t$ be the set of terminals departing. Let $A_t = A_t \setminus D_t$. Let $T_t = \cup_{v \in A_t} p_v$.
   - (Tree Follow Moves.) While $T_t$ is not in equilibrium: Use Algorithm select tree move to determine a tree-follow move to implement in $T_t$; let this be a move from $u$ to $v$, and let parent$(u)$ denote the parent of $u$ in $T_t$. Implement the sequence of improving moves for this tree-follow move to obtain the new routing tree $T_t \leftarrow T_t \setminus (u, \text{parent}(u)) \cup (u, v)$.

ACM Transactions on Economics and Computation, Vol. 9, No. 2, Article 11. Publication date: May 2021.
4.2 A Classification of States and Tree-follow Moves

Henceforth, we will assume that the algorithm maintains a family $\Pi = \{P_j\}_{j \in \mathbb{Z}}$ of partitions, where $P_j$ is a level-$j$ dual. This can be constructed in an online fashion as discussed in Appendix A. We classify the tree routings reachable via EQ-P dynamics into one of four states, depending on the charging structure defined by the solution against the family $\Pi$. We remark that not all tree routings are reachable via EQ-P dynamics; indeed, even the set of equilibria obtained is smaller than the set of all equilibria. Let $T$ be a routing tree for some set of active terminals $A$. We say a vertex $u$ is a leaf (non-leaf) if it is a leaf (non-leaf) in $T$. Note that all leaves must be terminals, but a non-leaf vertex may or may not be a terminal.

(1) **Balanced-equilibrium**: In this state, no terminal (and therefore, no non-terminal vertex in $T$) has an improving move. Furthermore, every cut is charged at most once. (Note that not every NE is a balanced-equilibrium state.)

(2) **Balanced**: In this state, some terminals (and potentially non-terminals) may have improving moves, but every cut is charged at most once.

(3) **Leaf-unbalanced**: In this state, every cut is charged by at most one non-leaf vertex (and any number of leaf terminals). (Recall that leaf vertices in the routing tree are necessarily terminals.)

(4) **Non-leaf-unbalanced**: In this state, all but one of the cuts are charged by at most one non-leaf vertex (and any number of leaf terminals). The exceptional cut, which we denote by $S^*$, is charged by at most two non-leaf vertices, say, $u$ and $v$ (and any number of leaf terminals). One of these, $u$ or $v$, must be the last vertex to have made a (tree-follow) move.

Note: balanced-equilibrium $\subseteq$ balanced $\subseteq$ leaf-unbalanced $\subseteq$ non-leaf-unbalanced, where $A \subseteq B$ implies that a routing tree in state $A$ is also in state $B$.

**Selecting a Tree-Follow Move.** To define the tree-follow move performed in a non-equilibrium tree state $T$, we establish a system of priorities among the improving tree moves based on the current state of the routing tree. A tree follow move of $u$ to $v$ is said to be a leaf move if $v$ is a leaf in $T$, and a non-leaf move otherwise. Generally speaking, non-leaf moves are prioritized over all leaf moves, i.e., if a non-leaf move is available for $u$, then it will not perform a leaf move. Next, we define the individual rules of priority within the two classes: non-leaf and leaf moves. For non-leaf moves, the move with the minimum cost of the $(u, v)$ parent edge has the highest priority. However, for leaf moves, if there is a leaf terminal $v$ such that $u$ and $v$ are currently charging the same dual, then $u$’s move to $v$ has highest priority, breaking ties arbitrarily among all such $v$. All other leaf moves have the same priority, i.e., if there are no moves of the above kind, then an arbitrary leaf move is chosen. Formally, the following algorithm selects which move to make in any given state.

| **Algorithm** SELECT TREE MOVE |
|-----------------------------|
| (1) **Balanced-equilibrium**: No terminal has an improving move. The system can deviate from an equilibrium state only via arrivals or departure events. |
| (2) **Balanced**: In this state, for any vertex $u$ that has an improving tree move, move $u$ to the closest vertex to which it has an improving move. |
| (3) **Leaf-unbalanced**: |
| (a) If there exists a leaf terminal $u$ with a non-leaf move, then make any such move for $u$. |
| (b) Else, if there exists a non-leaf vertex $u$ with a non-leaf move, then move $u$ to the closest such non-leaf $v$. |
Timing Matters: Online Dynamics in Broadcast Games

Fig. 3. This figure illustrates a state where two nodes $u$ and $v$ both charge the same dual cut at level $i$. If one of these vertices ($v$) is a leaf vertex, then either $u$ or $v$ has an improving move. $v$’s improving move is illustrated as the red path.

(c) Else, if there exists a non-leaf vertex $u$ and a leaf terminal $v$ such that $u$ and $v$ are charging the same dual cut, then move $u$ to $v$. If there are multiple such leaf terminals $v$, then make any such move.

(d) Else, make any improving move. (This will necessarily be a leaf-to-leaf move by exclusion of the previous three cases.)

(4) non-leaf-unbalanced: Let $u$ and $v$ be the non-leaf vertices that are charging the special cut $S^*$. If $u$ has an improving move to $v$, then move $u$, else move $v$, in either case to the closest vertex to which they have an improving move.

The validity of the algorithm depends on two claims. The first (Lemma 4.2) shows that whenever a cut is being charged by a leaf and a non-leaf, at least one of these two vertices has an improving move to the other. In this case, we can find a valid tree-move for Step (3c) of select tree move. The second (Lemma 4.4) shows that in a non-leaf-unbalanced state, whenever a cut is being charged by two non-leaves, at least one of these two vertices has an improving move to the other; we can then find a valid tree-move for Step (4) of select tree move.

Each of these claims hinges upon observing that if two vertices charge the same dual cut and one of them is a leaf, then one of the two has an improving move. In particular, the parent edges of these vertices are much longer than the distance between the two. Figure 3 below illustrates the situation. The blue paths represent the current strategies of the two nodes, and the red path represents a potential improving move for $v$. The following lemmas provide a formal analysis:

LEMMA 4.2. Let the routing tree $T$ be in non-leaf-unbalanced state but not in balanced state. Let $u, v$ be a pair of vertices in $T$ charging the same dual cut $D$ at some level $i$, where at most one of $u$ or $v$ is a non-leaf vertex in $T$. Then, at least one of $u$ moving to $v$ and $v$ moving to $u$ is an improving move.

Proof. Let $p_u$ and $p_v$ be the routing paths of $u$ and $v$, respectively, where $u$’s parent edge is $(u, x)$ and $v$’s parent edge is $(v, y)$. Then, we have $c_{ux}, c_{vy} \in [2^{i+2}, 2^{i+3})$. Moreover, since $u$ and $v$ belong to the same cut at level $i$, $c_{uv} < 2^i$. Therefore,

$$c_{ux}, c_{vy} > 4c_{uv}. \quad (1)$$

Let $N_e$ be the number of vertices using edge $e$. Since at most one of $u$ or $v$ is a non-leaf in $T$, we can assume w.l.o.g. that $u$ is a leaf in $T$ (otherwise, the contradiction that we derive below for edge $(u, x)$ can be derived instead for edge $(v, y)$). In other words, $N_{ux} = 1$. 

ACM Transactions on Economics and Computation, Vol. 9, No. 2, Article 11. Publication date: May 2021.
For the sake of contradiction, let us assume that neither \( u \) moving to \( v \), nor \( v \) moving to \( u \) is an improving move. Since \( u \) moving to \( v \) is not an improving move, we have the shared cost on path \( p_u \) is at most the shared cost for \( u \) if it shifts to the path \((u, v) \cup p_v\). Formally, we have

\[
\begin{align*}
&c_{ux} + \sum_{e \in p_u \setminus (u, x)} \frac{c_e}{N_e} \leq c_{uv} + \sum_{e \in p_v \setminus (u, x)} \frac{c_e}{N_e} + \sum_{e \in p_u \cap p_v} \frac{c_e}{N_e}, \\
&\text{i.e., } c_{ux} + \sum_{e \in p_u \setminus (u, x)} \frac{c_e}{N_e} \leq c_{uv} + \sum_{e \in p_v \setminus (u, x)} \frac{c_e}{N_e} + \sum_{e \in p_u \cap p_v} \frac{c_e}{N_e},
\end{align*}
\]

Similarly, since \( v \) moving to \( u \) is not an improving move, we have

\[
\begin{align*}
&\sum_{e \in p_v \setminus (u, x)} \frac{c_e}{N_e} + \sum_{e \in p_v \setminus (u, x)} \frac{c_e}{N_e} \leq c_{uv} + \sum_{e \in p_u \setminus (u, x)} \frac{c_e}{N_e} + \sum_{e \in p_u \setminus (u, x)} \frac{c_e}{N_e}, \\
&\text{i.e., } \sum_{e \in p_v \setminus (u, x)} \frac{c_e}{N_e} \leq c_{uv} + \sum_{e \in p_u \setminus (u, x)} \frac{c_e}{N_e} + \sum_{e \in p_u \setminus (u, x)} \frac{c_e}{N_e},
\end{align*}
\]

Adding inequalities (2) and (3), and replacing \( N_{ux} = 1 \), we get

\[
\begin{align*}
&c_{ux} + \sum_{e \in p_u \setminus (u, x)} \frac{c_e}{N_e} + \sum_{e \in p_v \setminus (u, x)} \frac{c_e}{N_e} \leq c_{uv} + \sum_{e \in p_u \setminus (u, x)} \frac{c_e}{N_e} + \sum_{e \in p_v \setminus (u, x)} \frac{c_e}{N_e} + c_{uv} + c_{ux} + \sum_{e \in p_u \setminus (u, x)} \frac{c_e}{N_e} + \sum_{e \in p_v \setminus (u, x)} \frac{c_e}{N_e}, \\
&\text{i.e., } c_{ux} \leq 4c_{uv},
\end{align*}
\]

which contradicts inequality (1). Thus, it must be the case that at least one of \( u \) moving to \( v \) or \( v \) moving to \( u \) is an improving move. \( \square \)

We now show a stability property of the improving moves made by Algorithm select tree move, namely, that moving a terminal \( u \) does not create any new improving moves for it. We will then be ready to prove Lemma 4.4.

**Lemma 4.3.** Suppose that in the current routing tree, vertex \( u \) moving to vertex \( v \) is not an improving move for \( u \). Then, after \( u \) moves to some other vertex \( x \) (using a tree-follow move), \( u \) moving to \( v \) is still not an improving move.

**Proof.** Let \( p_u \) and \( p_v \) denote the paths connecting \( u \) and \( v \), respectively, to \( r \) in the routing tree \( T \) after \( u \)’s move. Let \( N_e \) and \( N'_e \) be the number of terminals, except \( u \), that are routing through edge \( e \) before and after \( u \)’s tree move, respectively. For all edges in \( p_u \), we have \( N'_e \geq N_e \); for all other edges in \( T \), \( N'_e \leq N_e \). Since \( u \) moving to \( v \) was not an improving move but moving to \( x \) was, the shared cost of \( u \) if it moved to \( v \) would have been more than its shared cost after the tree move...
to \( p_u \). Hence,

\[
\begin{align*}
&c_{uv} + \sum_{e \in P_u} \frac{c_e}{N_e + 1} > \sum_{e \in P_u} \frac{c_e}{N_e + 1}, \\
\text{i.e.,} & \quad c_{uv} + \sum_{e \in P_u \cup P_v} \frac{c_e}{N_e + 1} > \sum_{e \in P_u \cup P_v} \frac{c_e}{N_e + 1}, \\
\text{i.e.,} & \quad c_{uv} + \sum_{e \in P_u \cup P_v} \frac{c_e}{N_e'} + \sum_{e \in P_u \cap P_v} \frac{c_e}{N_e' + 1} > \sum_{e \in P_u \cup P_v} \frac{c_e}{N_e' + 1}, \\
\text{i.e.,} & \quad c_{uv} + \sum_{e \in P_u \cup P_v} \frac{c_e}{N_e'} > \sum_{e \in P_u \cup P_v} \frac{c_e}{N_e'} + 1.
\end{align*}
\]

Hence, \( u \) moving to \( v \) is not an improving move after \( u \)'s move to \( x \).

**Lemma 4.4.** Let \( T \) be a routing tree in a non-leaf-unbalanced but not a leaf-unbalanced state, arrived at by implementing one of the steps (2)–(4) in Algorithm select tree move. Let \( u \) and \( v \) be the non-leaf vertices charging the special cut \( S^* \). Then either \( u \) to \( v \) or \( v \) to \( u \) is an improving move.

**Proof.** By the definition of the non-leaf-unbalanced state, it must be the case that either \( u \) or \( v \) was the last to start a tree-follow move. Without loss of generality, say that \( u \)'s last move was \((u, x)\) and \((v, y)\) are, respectively, the parent edges of \( u \) and \( v \). Then, we have \( c_{uv}, c_{vy} \in [2^{i+2}, 2^{i+3}) \). Moreover, \( c_{uv} < 2^i \), since \( u \) and \( v \) both belong to the cut \( S^* \) at level \( i \). Then, we have

\[
c_{uv}, c_{vy} > 4c_{uv}.
\]

(4)

We first claim that prior to \( u \)'s move to \( x \), \( u \) did not have an improving move to \( v \). Suppose, for contradiction, that \( u \) did have an improving move to \( v \). Then, since \((u, v)\) is shorter than \((u, x)\), and \( u \) and \( v \) are non-leaves, \( u \)'s potential move would have triggered Steps (2), (3b), or (4). In each of these cases, \( u \) would have preferred the move to the closer vertex \( v \) over the move to \( x \). It follows that \( u \) to \( v \) was not an improving move before \( u \)'s move to \( x \). Lemma 4.3 ensures that \( u \) moving to \( v \) is not an improving move after \( u \)'s move to \( x \) either. We therefore have,

\[
\begin{align*}
&c_{ux} + \sum_{e \in P_u \setminus (u, x)} \frac{c_e}{N_e} \leq c_{uv} + \sum_{e \in P_u \setminus P_u} \frac{c_e}{N_e + 1} + \sum_{e \in P_u \cap P_u} \frac{c_e}{N_e}. \\
\text{i.e.,} & \quad c_{ux} + \sum_{e \in (P_u \setminus (u, x)) \setminus P_u} \frac{c_e}{N_e} \leq c_{uv} + \sum_{e \in P_u \setminus P_u} \frac{c_e}{N_e + 1} + \sum_{e \in P_u \cap P_u} \frac{c_e}{N_e}, \\
\text{i.e.,} & \quad c_{ux} + \sum_{e \in (P_u \setminus (u, x)) \setminus P_u} \frac{c_e}{N_e} \leq c_{uv} + \sum_{e \in P_u \setminus P_u} \frac{c_e}{N_e + 1}.
\end{align*}
\]

(5)
The rest of the proof is devoted to showing that \( v \) moving to \( u \) is an improving move. Let us assume for the sake of contradiction that \( v \) moving to \( u \) is not an improving move either. Then, we have

\[
\sum_{e \in p_v} \frac{c_e}{N_e} \leq c_{uv} + \sum_{e \in p_u \setminus p_v} \frac{c_e}{N_e} + \sum_{e \in p_u \cap p_v} \frac{c_e}{N_e},
\]

i.e.,

\[
\sum_{e \in p_u \setminus p_v} \frac{c_e}{N_e} + \sum_{e \in p_u \cap p_v} \frac{c_e}{N_e} \leq c_{uv} + \sum_{e \in p_u \setminus p_v} \frac{c_e}{N_e} + \sum_{e \in p_u \cap p_v} \frac{c_e}{N_e}.
\]

Adding the inequalities (5) and (6), and observing that \( N_{ux} \geq 1 \), we get

\[
c_{ux} + \sum_{e \in (p_u / (u, x)) \setminus p_v} \frac{c_e}{N_e} \leq c_{uv} + \sum_{e \in p_u \setminus p_v} \frac{c_e}{N_e} + c_{uv} + \sum_{e \in p_u \cap p_v} \frac{c_e}{N_e} + \frac{c_{ux}}{N_{ux} + 1}.
\]

i.e.,

\[
c_{ux} \leq 2c_{uv} + \frac{c_{ux}}{2},
\]

which contradicts inequality (4). Thus, it must be the case that \( v \) moving to \( u \) is an improving move. \( \square \)

5 ANALYSIS OF EQ-P DYNAMICS

We now give a proof of Theorem 1.2. Our argument hinges on a closure property: The epoch starts with the routing tree being in the balanced-equilibrium state; Lemma 5.3 argues that whenever the current routing tree is not in equilibrium, at least one improving move exists, and we can use Algorithm select tree move to make a move; Lemma 5.4 then shows that for the moves made by Algorithm select tree move, the routing tree remains in one of the four states defined above, in particular, it is always in a non-leaf-unbalanced state. The epoch ends when the routing tree re-enters a balanced-equilibrium state. At this point, by definition, each dual cut is charged at most once, and therefore, by Lemma 2.3, the cost of the routing tree is bounded and Theorem 1.2 follows. We must also argue termination of the sequence of moves, but this follows directly from a standard potential argument based on the fact that all our moves are improving moves.

The arrival or departure phase. We first consider the situation where the epoch begins with an arrival event. Every arriving terminal \( u \) chooses its best response path \( p_u^* \) as its current routing path \( p_u \). We claim that the new routing paths \( p_u \mid u \in U_t \) along with the current routing tree \( T \) continue to form a tree solution and we end up in a leaf-unbalanced state.

Lemma 5.1. Suppose a set of new terminals \( U_t \) arrives in epoch \( t \) when the routing paths of the existing terminals are in an equilibrium state. Then, for each new terminal \( u \in U_t \), the chosen routing path \( p_u \) comprises a single edge \((u, v)\) connecting \( u \) to an existing vertex \( v \) in the current routing tree \( T \), and then following the unique path in \( T \) from \( v \) to \( r \).

Proof. The lemma has two parts:

- \( p_u \) has a single edge \((u, v)\) connecting \( u \) to \( v \), and
- the path \( p_u \) does not deviate from \( T \) between \( v \) and \( r \).
The first statement is a direct consequence of our tie-breaking rule for best response paths: $u$ is the only terminal using the portion of $p_u$ from $u$ to $T$; if this segment consists of a multi-hop path from $u$ to some vertex $v \in T$, then short-cutting this segment and using the direct $(u, v)$ edge instead is potentially cheaper and has fewer edges with only $u$ using them.

For the sake of contradiction, suppose the second statement is false. Then, we claim that the vertex $v$ has an improving move. This contradicts the fact that the routing paths are in equilibrium when terminal $u$ arrives. To prove the claim, suppose first that $v$ is a terminal. Let $p_v$ denote the path along $T$ from $v$ to the root, and let $q_v$ denote the segment of $p_u$ from $v$ to the root. Let $E_v$ be the set of edges in $p_v \setminus q_v$, and $E_u$ the set of edges in $q_v \setminus p_u$. Let $N_e$ denote the number of terminals using edge $e$ prior to any arrivals in this epoch. Since $u$ follows its best response path, we have that

$$\sum_{e \in E_u} \frac{c_e}{N_e + 1} \leq \sum_{e \in E_v} \frac{c_e}{N_e + 1}.$$  

The cost share of $v$ over edges in $E_v$ is $\sum_{e \in E_v} \frac{c_e}{N_e}$, whereas over edges in $E_u$ if $v$ were to switch to taking the path $q_v$ would be $\sum_{e \in E_u} \frac{c_e}{N_e + 1}$. From the above inequality, it follows that

$$\sum_{e \in E_u} \frac{c_e}{N_e + 1} < \sum_{e \in E_v} \frac{c_e}{N_e},$$

which implies that $v$’s cost share would improve strictly by switching from $p_v$ to $q_v$. When $v$ is not a terminal, but is on the current path of some other terminal $w$, an identical argument shows that $w$ (and therefore $v$) has an improving move. \hfill $\square$

Using the above lemma, we can now claim the following:

**Lemma 5.2.** After the arrival or departure of a set of terminals in an balanced-equilibrium state, the routing tree $T$ remains in a leaf-unbalanced state.

**Proof.** For an arrival event, this is a direct consequence of Lemma 5.1. Since every new terminal is a leaf, there is at most one non-leaf vertex charging every cut. After a set of terminal departures, the routing tree clearly remains in a balanced state, since charges to dual cuts can only decrease. \hfill $\square$

**Sequence of tree-follow moves.** We now consider improving moves made by Algorithm **select tree move.** We first observe that for any routing tree that is not at equilibrium, there must exist an improving tree-follow move.

**Lemma 5.3.** If the routing tree is not in equilibrium, then at least one improving tree-follow move exists.

**Proof.** Let the current routing path of every terminal $x$ be denoted $p_x$ and the union of these paths be tree $T$. For notational convenience, we also denote the path in the routing tree from a non-terminal vertex $x$ to $r$ by $p_x$. We first identify a vertex $w$ that has an improving path $q_w$ that contains exactly one arc not in $T$. Since $T$ is not an equilibrium, there exists a terminal $x$ whose best response path $p_x^*$ differs from $p_x$. Let $(u, v)$ be the edge that is closest to $r$ on $p_x^*$ and is not in $T$, with $v$ closer to $r$ than $u$. Let $w$ be any terminal in the subtree of $T$ rooted at $u$. We claim the path $q_w$ formed by taking the subpath in $T$ from $w$ to $u$ and then the subpath of $p_x^*$ from $u$ to $r$ is an improving path for $w$. Indeed, if its shared cost is at least the shared cost of $p_w$, then we can find a path for $x$ whose shared cost is at most the shared cost of $p_x^*$. Consider the path $q_x$ for $x$ formed by taking the subpath $p_x^*$ from $x$ to $u$ and then taking the subpath from $u$ to $r$. The difference in shared cost for $x$ between $q_x$ and $p_x^*$ equals the difference in shared cost for $w$ in $p_w$ and $q_w$. Due to
our tie-breaking rule, the best response path \( p_x^* \) must be strictly better than \( q_x \), since \( q_x \) has fewer new edges. Thus, \( q_w \) must be strictly better than \( p_w \) as desired.

Now let \( T' = T \setminus (u, \text{parent}(u)) \cup (u, v) \) where \( \text{parent}(u) \) is the parent of \( u \) in \( T \). We now give a series of improving moves that give us the routing tree \( T' \). Order the terminals in subtree of \( T \) rooted at \( u \). For each such terminal \( w \) change its path from \( p_w \) to \( q_w \) as defined above. It is clear that the union of all routing paths is exactly \( T' \). It remains to show that they are all improving paths for their respective terminals. From the above argument, for the first terminal \( w_1 \), \( q_{w_1} \) is clearly an improving path over \( p_{w_1} \) in \( T \). For any latter terminal \( w \) in the sequence, the shared cost of \( q_w \) has only reduced as compared to its shared cost in the solution \( T \), since more terminals are using the edges on the subpath from \( u \) to \( x \). Moreover, the shared cost of \( p_w \) has only increased as compared to its shared cost in \( T \), as fewer terminals are using the subpath of \( w \) from \( u \) to \( r \). The subpath from \( w \) to \( u \) remains identical and same number of terminals keep using it and therefore, \( q_w \) remains an improving path in the intermediate solution as well for each of the terminals in the sequence. \( \square \)

We therefore have the following simple observation:

**Observation 1.** In EQ-P dynamics the routing paths at the end of a phase always form a tree.

We are now ready to establish our main technical result of this section, namely, that the four states defined in Section 4 are closed under EQ-P dynamics. The proof proceeds via a detailed case analysis.

**Lemma 5.4.** Let \( T \) be the routing tree for which we make an improving tree-move in Step (3) of Algorithm select tree move.

(i) Suppose \( T \) is in a balanced state but not in a balanced-equilibrium state, then after the move selected in Step (2) of select tree move, the resulting tree is in a non-leaf-unbalanced state.

(ii) Suppose \( T \) is in a leaf-unbalanced state, then after the move selected in Step (3) of select tree move, the resulting tree is in a non-leaf-unbalanced state.

(iii) Suppose \( T \) is in a non-leaf-unbalanced state, then after the move selected in Step (4) of select tree move, the resulting tree is in a non-leaf-unbalanced state.

**Proof.** We complete the proof by a detailed case analysis.

(i) Let \( T \) be in a balanced state but not a balanced-equilibrium. Therefore, every dual is being charged at most once in \( T \). After the move, the new tree \( T' \) contains exactly one edge not in \( T \). The charging for this edge can introduce at most one dual cut that is charged more than once. Thus, the new routing tree is in a non-leaf-unbalanced state.

(ii) Let \( T \) be a tree in a leaf-unbalanced state. We now consider different cases depending on the move chosen by Algorithm select tree move.

**Step (3a):** Suppose the algorithm makes a leaf to non-leaf move in Step (3a), then the only new edge introduced is \( u \)'s parent edge. Since \( u \) remains a leaf in \( T \) and its new parent was already a non-leaf vertex, no new non-leaf vertex is introduced. Thus, in the new routing tree, every dual is charged by at most one non-leaf vertex. This implies that the leaf-unbalanced state is preserved.

**Step (3b):** Suppose there is no improving move in Step (3a), and algorithm makes a non-leaf to non-leaf move in Step (3b) for vertex \( u \). Let the new edge introduced be \((u, v)\) where \( v \) is a non-leaf vertex. After this move, the only cut that has an additional non-leaf vertex charging to it is the cut being charged by \( u \) (call it \( S \)). Prior to this move, \( S \) had at least one non-leaf vertex charging it. After \( u \)'s move, it has at most two non-leaves charging it, with one of them \((u)\)
having made the last move. Therefore, $T$ is in a LEAF-UNBALANCED or NON-LEAF-UNBALANCED state.

**Step (3c):** Suppose there are no improving moves in Steps (3a) and (3b), and the algorithm performs a non-leaf to leaf move in Step (3c) from $u$ to $v$. Prior to the move, $u$ and $v$ were charging the same cut, say, $S$ at level $i$. After $u$’s move, $u$’s parent edge, $(u, v)$, is of length $< 2^i$, and therefore, $u$ charges a cut different from $S$ (call it $S'$), whereas $v$ continues to charge $S$.

Therefore, the only cuts that get charged by new non-leaves after $u$’s move are $S$ and $S'$. $S$ was previously being charged by a single non-leaf, namely, $u$; Now it is charged by only one non-leaf, namely $v$. $S'$ was previously being charged by at most one non-leaf (by virtue of the routing tree being in a LEAF-UNBALANCED state), so now it is being charged by at most two non-leaves, one of which is $u$. Therefore, $T$ is in a LEAF-UNBALANCED or NON-LEAF-UNBALANCED state.

**Step (3d):** Finally, consider the scenario where there are no improving moves in any of Steps (3a), (3b), and (3c). In this case, the algorithm makes a leaf to leaf move in Step (3d) from $u$ to $v$. The only new non-leaf vertex created by the move is $v$. We first argue that the dual charged by $v$ (say, $S$) does not have a second non-leaf vertex charging it. Since Step (3c) was not executed, it follows that no cut was being charged by both a non-leaf and a leaf vertex before the move (although multiple leaf terminals might be charging the same cut). In particular, $v$ was a leaf vertex charging $S$ before the move, and so, no other non-leaf vertex was charging $S$ before or after the move.

The only other cut that gets a new charge after the move is the cut charged by $u$’s new edge. Since $u$ is a leaf, this cut continues to have at most one non-leaf charging it. Therefore, the routing tree is in a LEAF-UNBALANCED state.

(iii) Let $T$ be in a NON-LEAF-UNBALANCED state. Recall from the definition of Step (4) that $u$ and $v$ are the two non-leaf vertices charging the special cut $S^*$ at some level $i$, and $u$ has an improving move to $v$. Then, we have $c_{uw} < 2^i$. Suppose that $u$ made the improving move to $w$, where $w$ can be $v$. Since $u$ chooses the improving move to the closest vertex, we have $c_{uw} \leq c_{uv} < 2^i$. After the move, $u$ must charge a cut, say, $S$, whose level is strictly less than $i$. Thus, $S \neq S^*$, and $S^*$ now has only a single non-leaf charging it. Moreover, before the move $S$ had at most one non-leaf charging it. Now, along with $u$, it can have two non-leaves charging it but one of them, $u$, has made the last move. Thus, the new tree is in a NON-LEAF-UNBALANCED state. □

**APPENDICES**

A  CONSTRUCTING THE DUAL IN AN ONLINE FASHION

The classification of tree routings described in Section 4 as well as the description of algorithm SELECT TREE MOVE relies on the knowledge of the dual family $\Pi$ defined in Section 4 to which we charge the cost of our solution. If the underlying graph $G$ is known in advance, there are standard techniques for constructing a family of duals with the desired properties. In our setting, the underlying graph may not be known in advance and may instead be revealed over time as terminals arrive. We now describe a simple greedy procedure for constructing the dual family in an online fashion.

Recall that the graph $G$ is complete and edge lengths form a metric, i.e., they satisfy the triangle inequality. We use $c(u, v)$ to denote the length of the edge between vertices $u$ and $v$. The graph
is revealed vertex-by-vertex, and every time a vertex is added, all edges between that vertex and previously added vertices are revealed. The level $j$ dual for any integer $j$ is constructed as follows: At any point of time, we have some number of components $S_1, \ldots, S_m$ in the dual, with centers $s_1, \ldots, s_m$, respectively. At the beginning when the graph contains a single vertex, we have a single component with that vertex as its center. When the next vertex, say, $v_i$, arrives, if there exists a center $s_i$ with $c(s_i, v) < 2^{j-1}$, then we add $v$ to the $i$th component $S_i$. Otherwise, we create a new component $S_{m+1}$ with center $s_{m+1} = v$. By construction, it holds that every vertex in component $S_i$ is at a distance less than $2^{j-1}$ from its center $s_i$, and therefore, the diameter of the component is less than $2^j$. Also, by construction, the distance between any two centers is at least $2^{j-1}$. Therefore, the constructed dual satisfies the properties listed in Section 4.

B BAD EXAMPLES

In Figure 4(a), we give an instance from Reference [4] where the price of anarchy is arbitrarily large. In the above example, there are $n + 1$ agents at $u$. If the solution picks the edge of weight $n$ instead of the edge of weight 1, a simple check shows that it is still in equilibrium. Since the optimal solution picks the edge of weight 1 instead of weight $n$, the price of anarchy is $n$.

![Fig. 4. Bad Examples for Price of Anarchy and comparison with Minimum Steiner Tree.]

In Figure 4(b), we give an instance where the natural dynamics leads to solution that is much more expensive than the minimum Steiner tree. Consider the following sequence of arrivals where each agent picks the best response path on arrival. First $n$ agents arrive on $v_{n-1}$, then $n$ agents arrive on $v_{n-2}$, and so on. Finally, $n$ agents arrive on $u$. Observe that in each phase of $n$ arrivals, the best response dynamics introduces the edge $(v_i, v_{i+1})$ and thus the solution at the end is the long path from $u$ to $r$. Now, all the agents, except the $n$ agents at $u$, depart. Observe that the solution is still in equilibrium, since it is identical to the price of anarchy solution in Figure 4. But the weight of minimum Steiner tree is 1 for the agents that survive. This shows that the dynamics can lead to a much costlier solution as compared to the minimum Steiner tree. A more apt comparison is to the cost of the minimum spanning tree over all of the arriving clients. In this case, the minimum spanning tree costs $n$, which is exactly our solution.

ACKNOWLEDGMENTS

Part of this work was done when all the authors were visiting Microsoft Research - Redmond.

REFERENCES

[1] Sebastian Aland, Dominic Dumrauf, Martin Gairing, Burkhard Monien, and Florian Schoppmann. 2011. Exact price of anarchy for polynomial congestion games. SIAM J. Comput. 40, 5 (2011), 1211–1233.

[2] Susanne Albers. 2009. On the value of coordination in network design. SIAM J. Comput. 38, 6 (2009), 2273–2302.

[3] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. 2006. A general approach to online network optimization problems. ACM Trans. Algor. 2, 4 (2006), 640–660.
[4] Elliot Anshelevich, Anirban Dasgupta, Jon M. Kleinberg, Éva Tardos, Tom Wexler, and Tim Roughgarden. 2008. The price of stability for network design with fair cost allocation. *SIAM J. Comput.* 38, 4 (2008), 1602–1623.

[5] Baruch Awerbuch, Yossi Azar, and Yair Bartal. 2004. On-line generalized Steiner problem. *Theor. Comput. Sci.* 324, 2–3 (2004), 313–324.

[6] Maria-Florina Balcan, Avrim Blum, and Yishay Mansour. 2013. Circumventing the price of anarchy: Leading dynamics to good behavior. *SIAM J. Comput.* 42, 1 (2013), 230–264.

[7] René Beier, Artur Czumaj, Piotr Krysta, and Berthold Vöcking. 2006. Computing equilibria for a service provider game with (im)perfect information. *ACM Trans. Algor.* 2, 4 (2006), 679–706.

[8] Piotr Berman and Chris Coulston. 1997. On-line algorithms for Steiner tree problems. In *Proceedings of the 29th ACM Symposium on Theory of Computing*. ACM, 344–353.

[9] Vittorio Bilò and Roberta Bove. 2011. Bounds on the price of stability of undirected network design games with three players. *J. Interconnect. Netw.* 12, 1–2 (2011), 1–17.

[10] Vittorio Bilò, Michele Flammini, and Luca Moscardelli. 2013. The price of stability for undirected broadcast network design with fair cost allocation is constant. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS’13)*. 638–647.

[11] Moses Charikar, Howard J. Karloff, Claire Mathieu, Joseph Naor, and Michael E. Saks. 2008. Online multicasting with egalitarian cost sharing. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA’08)*. 70–76.

[12] Chandra Chekuri, Julia Chuzhoy, Liane Lewin-Eytan, Joseph (Seffi) Naor, and Ariel Orda. 2007. Non-cooperative multicast and facility location games. *IEEE J. Select. Areas Commun.* 25, 6 (2007), 1193–1206.

[13] Ho-Lin Chen and Tim Roughgarden. 2009. Network design with weighted players. *Theor. Comput. Syst.* 45, 2 (2009), 302–324.

[14] George Christodoulou, Christine Chung, Katrina Ligett, Evangelia Pyrga, and Rob van Stee. 2009. On the price of stability for undirected network design. In *Approximation and Online Algorithms*, 7th International Workshop, WAOA 2009, Copenhagen, Denmark, September 10-11, 2009. Revised Papers. Springer, 86–97.

[15] Alina Ene, Deeparnab Chakrabarty, Ravishankar Krishnaswamy, and Debmalya Panigrahi. 2015. Online buy-at-bulk network design. In *Proceedings of the IEEE 56th Symposium on Foundations of Computer Science (FOCS’15)*. 545–562.

[16] Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. 2004. The complexity of pure Nash equilibria. In *Proceedings of the 36th Symposium on Theory of Computing*. 604–612.

[17] Angelo Fanelli, Dariusz Leniowski, Gianpietro Monaco, and Piotr Sankowski. 2015. The ring design game with fair cost allocation. *Theor. Comput. Sci.* 562 (2015), 90–100.

[18] Amos Fiat, Haim Kaplan, Meital Levy, Svetlana Olonetsky, and Ronen Shabo. 2006. On the price of stability for designing undirected networks with fair cost allocations. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP’06)*. 608–618.

[19] Rupert Freeman, Samuel Haney, and Debmalya Panigrahi. 2016. On the price of stability of undirected multicast games. In *Web and Internet Economics - 12th International Conference, WINE 2016, Montreal, Canada, December 11-14, 2016, Proceedings*. Springer, 354–368.

[20] Anupam Gupta, R. Ravi, Kunal Talwar, and Seun William Umboh. 2017. LAST but not least: Online spanners for buy-at-bulk. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19. SIAM, 589–599.

[21] MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. 2013. Online node-weighted Steiner forest and extensions via disk paintings. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS’13)*. 558–567.

[22] MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. 2014. Near-optimal online algorithms for prize-collecting Steiner problems. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP’14)*. 576–587.

[23] Tobias Harks and Max Klimm. 2012. On the existence of pure Nash equilibria in weighted congestion games. *Math. Oper. Res.* 37, 3 (2012), 419–436.

[24] Makoto Imase and Bernard M. Waxman. 1991. Dynamic Steiner tree problem. *ACM Trans. Atron.* 2, 4 (2006), 679–706.

[25] Yasushi Kawase and Kazuhisa Makino. 2013. Nash equilibria with minimum potential in undirected broadcast games. *Theor. Comput. Sci.* 482 (2013), 33–47.

[26] Euiwoong Lee and Katrina Liggett. 2013. Improved bounds on the price of stability in network cost sharing games. In *Proceedings of the ACM Conference on Economics and Computation (EC’13)*. 607–620.

[27] Jian Li. 2009. An O(log(n)/log(log(n))) upper bound on the price of stability for undirected Shapley network design games. *Inf. Proc. Lett.* 109, 15 (2009), 876–878.

[28] Igal Milchtaich. 1996. Congestion games with player-specific payoff function. *Games Econ. Behav.* 13 (1996), 111–124.

[29] Dov Monderer and Lloyd S. Shapley. 1996. Potential games. *Games Econ. Behav.* 14 (1996), 124–143.
[30] Joseph Naor, Debmalya Panigrahi, and Mohit Singh. 2011. Online node-weighted Steiner tree and related problems. In Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS’11). 210–219.

[31] Jiawei Qian, Seeun William Umboh, and David P. Williamson. 2018. Online constrained forest and prize-collecting network design. Algorithmica 80, 11 (2018), 3335–3364.

[32] Nissan Law-Yone Ron Holzman. 1997. Strong equilibrium in congestion games. Games Econ. Behav. 21 (1997), 85–101.

[33] Robert W. Rosenthal. 1973. A class of games possessing pure-strategy Nash equilibria. Int. J. Game Theor. 2, 1 (1973), 65–67.

[34] Vasili Syrgkanis. 2010. The complexity of equilibria in cost sharing games. In Internet and Network Economics. Springer Berlin, 366–377.

[35] Takashi Ui. 2000. A Shapley value representation of potential games. Games Econ. Behav. 31, 1 (2000), 121–135.

[36] Seeun Umboh. 2015. Online network design algorithms via hierarchical decompositions. In Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms. SIAM, 1373–1387.

Received April 2019; revised April 2020; accepted August 2020