Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 2206 – 2220
© TÜBİTAK
doi:10.55730/1300-0632.3934

On approximate Nash equilibria of the two-source connection game

Buğra ÇAŞKURLU[1,*], **Utku Umur AÇIKALIN**[1], **Fatih Erdem KIZILKAYA**[1], **Özgün EKİCİ**[2]

[1]Department of Computer Engineering, Faculty of Engineering,
TOBB University of Economics and Technology, Ankara, Turkey
[2]Department of Economics, Faculty of Business, Özyeğin University, İstanbul, Turkey

Received: 19.10.2021 • **Accepted/Published Online:** 28.07.2022 • **Final Version:** 28.09.2022

Abstract: The arbitrary-sharing connection game is prominent in the network formation game literature [1]. An undirected graph with positive edge weights is given, where the weight of an edge is the cost of building it. An edge is built if agents contribute a sufficient amount for its construction. For agent i, the goal is to contribute the least possible amount while assuring that the source node s_i is connected to the terminal node t_i. In this paper, we study the special case of this game in which there are only two source nodes. In this setting, we prove that there exists a 2-approximate Nash equilibrium that is socially optimal. We also consider the further special case in which there are no auxiliary nodes (i.e., every node is a terminal or source node). In this further special case, we show that there exists a $\frac{3}{2}$-approximate Nash equilibrium that is socially optimal. Moreover, we show that it is computable in polynomial time.

Key words: Algorithmic game theory, network formation games, connection game, approximate Nash equilibrium

1. Introduction

The game theory emerged in the 20th century as a mathematical discipline to analyze the interactions of self-interested agents. Before the advent of the Internet, game theory was not part of computer science research studies, except for a few artificial intelligence applications [2]. The advent of the Internet changed this situation dramatically. With applications such as online auctions [3], sponsored search [4], cryptocurrencies [5], and multi-agent robot systems [6], computer systems have become the main venue for game theory. Algorithms and game theory intertwined, giving birth to the field today known as *algorithmic game theory* [7].

A game instance $\mathcal{G}$ in normal form is a triplet (N, Σ, c): $N = \{1, 2, \dots, n\}$ is the set of agents (players), $\Sigma = \Sigma_1 \times \ldots \times \Sigma_n$ is the strategy space, and $c = (c_1, \dots, c_n)$ is the profile of agents' cost functions. In this specification, Σ_i is agent i's set of possible strategies, and $c_i : \Sigma \to \mathbb{R}_{\geq 0}$ is agent i's cost function (her negative "payoff function"). A strategy of agent i is denoted by σ_i. The profile of agents' strategies, denoted by $\sigma = (\sigma_1, \dots, \sigma_n) \in \Sigma$, is referred to as a *strategy profile*, or a *solution*. The *social cost* of σ is the sum of individual costs at σ. The solution for which the social cost is minimum is called the *socially-optimal solution*. A solution is a *Nash equilibrium* if, at that solution, no agent can reduce her cost by unilaterally changing her strategy [8]. In a game instance, there may be multiple Nash equilibria, and a socially-optimal solution is not necessarily a Nash equilibrium.

In game theory, two important questions are whether a Nash equilibrium always exists, if it does and if it is unique. Additionally, in algorithmic game theory, a Nash equilibrium's computational tractability, and

*Correspondence: bcaskurlu@etu.edu.tr

the quantification of its inefficiency level are also important questions. In the literature, as a measure of the inefficiency level, the ratio of the social cost of a Nash equilibrium outcome to the optimal social cost is commonly used. The maximum and the minimum values that this ratio takes in Nash equilibrium outcomes are called, in order, the *price of anarchy* [9] and the *price of stability* [10].

The influence of the Internet brought networks into the center stage of algorithmic game theory studies [11]. In what is called *network formation games*, researchers try to understand the structure and the properties of networks constructed by self-interested agents [12]. A network is often modeled as a graph with positive edge weights, where the weight of an edge is the cost of building (or purchasing) it. In a typical network formation game, each agent sits at a node and wants her quality of service (QoS) requirement to be met. To this end, certain edges must be built, and the costs of building edges are shared by agents. Agents try to minimize their incurred costs while assuring that their QoS requirements are satisfied. The two defining features of a network formation game are the nature of agents' QoS requirements and the protocol by which edge costs are shared.

It was indeed the economists who initiated the research on network formation games [13]. They used this type of game to model the creation of social and economic networks [14]. In these games, agents typically form local connections in the form of direct links with other agents, and each agent aims to pay as little as possible while achieving some goal. This type of model is also employed in computer science, for instance, when modeling the peering relations among autonomous systems [15–17].

In this paper, we consider the *connection game*, which is a prominent network formation model [1]. In a connection game, agent i's QoS requirement is the connectivity between two agent-specific nodes, the source node s_i and the terminal node t_i.[1] Multiple agents may share the same source node (i.e. for $i, j \in N$, we may have $s_i = s_j$), and multiple agents may share the same terminal node (i.e. for $i, j \in N$, we may have $t_i = t_j$). Two special cases of the connection game considered in the literature are the *multicast game* and the *broadcast game*. In the multicast game, also called the *single-source connection game*, all agents share the same source node s. That is, for all $i \in N$, we have $s_i = s$. A node is said to be *auxiliary* if it is not a source node or a terminal node of some agent. The broadcast game is the special case of the multicast game in which there are no auxiliary nodes. Notice that the graph of a broadcast game with n agents has at most $n+1$ nodes, and each node other than s is necessarily the terminal node of some agent.

In a network formation game, how much agents pay for the construction of an edge is determined by a *cost-sharing scheme*. The most popular cost-sharing scheme in the literature is *fair-sharing* [10]. In the fair-sharing connection game, a strategy of agent i is a *simple* $s_i - t_i$ path.[2] Therefore, Σ_i is the set of all simple $s_i - t_i$ paths in the graph. For a solution $\sigma = (\sigma_1, \ldots, \sigma_n)$, if σ_i contains e, it means e is *selected* by agent i. An edge is built if it is selected by at least one agent and its cost is shared equally by those agents who selected it. The fair-sharing connection game falls into the class of "congestion games" for which the existence of Nash equilibrium is guaranteed [18]. The fair-sharing scheme is associated with some desirable properties. For instance, it eliminates the free-rider problem by ensuring that every agent pays for a fair portion of the cost of the edges that she uses. It can also be derived from the Shapley value [19]. For axiomatic studies on the fair-sharing scheme, see [19–21].

There is a major drawback of the fair-sharing scheme, however. In fair-sharing network formation games,

[1] In the original definition of the connection game [1], an agent's QoS requirement is the connectivity between a subset of nodes. The follow-up papers however consider the bare-bones case, as in our paper, in which the QoS requirement is the connectivity between a pair of nodes.

[2] A $v_0 - v_k$ path in a graph is a sequence of nodes $v_0, v_1, \ldots, v_k$ with the property that each consecutive pair v_i, v_{i+1} is joined by an edge. The path *contains* the edges $(v_0, v_1), \ldots, (v_{k-1}, v_k)$. A path is *simple* if all its nodes are distinct from one another.

the price of stability tends to be high. Anshelevich et al. [10] proved that in the fair-sharing connection game, the price of stability is bounded above by $H_n = \sum_{i=1}^{n} \frac{1}{i}$, and this bound is tight even in the special case of the broadcast game on directed networks.[3] On the other hand, establishing the price of stability on undirected networks is a long-standing open challenge. The best-known lower bounds for the price of stability, as of now, are 1.818 for the broadcast game, 1.862 for the multicast game, and 2.245 for the connection game [22]. The best-known upper bounds for the price of stability are $O(\lg n)$ for the connection game [10], $O(\frac{\lg n}{\lg \lg n})$ for the multicast game [23], and $O(1)$ for the broadcast game [24].

In this line of research, another key concept is the notion of an α-*approximate Nash equilibrium*. A solution σ is an α-approximate Nash equilibrium if no agent has a unilateral deviation that reduces her cost to less than $\frac{1}{\alpha}$ times her cost under σ. Another drawback of the fair-sharing scheme is that there may not exist any approximate Nash equilibrium that has a low social cost. See [25], for instance, for a fair-sharing network formation game where the socially-optimal solution is not an approximate Nash equilibrium for any α.

Another popular cost-sharing scheme in the literature is *arbitrary-sharing* [1]. In the arbitrary-sharing scheme, as their strategies, agents specify the amounts that they are willing to contribute for the construction of different edges. Thus, when there are m edges, agent i's strategy is a nonnegative vector of size m. An edge is built if the total amount contributed for its construction covers its cost, and it is not built otherwise. An arbitrary-sharing network formation game is not a congestion game, and in this game, the existence of a Nash equilibrium is not guaranteed. However, arbitrary-sharing network formation games are quite robust in having low-factor approximate Nash equilibrium solutions associated with low social cost [1, 26–28]. Arbitrary-sharing is also related to separable cost-sharing protocol design [29].

In their seminal paper, Anshelevich et al. [1] showed that the price of stability of the arbitrary-sharing multicast game is 1. In other words, there exists a Nash equilibrium that is socially optimal. This result also holds for directed networks. For the arbitrary-sharing broadcast game, the price of stability is 1 even for its "survivable" version, in which the QoS requirement for agent i is two edge-disjoint paths between s and t_i rather than a single path [26]. However, it is known that for the arbitrary-sharing connection game, a Nash equilibrium may not exist if there are two or more source nodes. Anshelevich et al. [1] proved the existence of a 3-approximate Nash equilibrium that is socially optimal.

The following real-life application motivates the network formation models under the arbitrary-sharing scheme. Imagine a huge nationwide road construction project in the US. The set of all potential roads to be built can be represented as an undirected graph. In this scenario, each continental state acts as an agent that aims to minimize the state's contributions while assuring that the state-specific connectivity requirements are satisfied. The strategy of each state is a nonnegative contribution toward the construction of each potential road to be built. The underlying game is an arbitrary-sharing network formation game. Notice that the social cost of the road network built may be significantly higher than that of the socially-optimal solution. In that setting, the federal government may act as the central authority that aims to ensure that the socially-optimal road network is built. The federal government, however, cannot directly control how states use their resources. But to ensure that the allocation of state resources leads to the construction of the socially-optimal network, the federal government may give subsidies for the construction of some roads in the network. However, the federal government would also be interested in minimizing the sum of the subsidies it gives. The theoretical results proving the existence of an α-approximate Nash equilibrium payment scheme that builds the socially-optimal

[3] Note that $H_n = \Theta(\lg n)$.

solution gives a theoretical upper bound on the total amount the federal government needs to spend. It is easy to see that in that case, the federal government can ensure the construction of the socially-optimal network by paying at most $\frac{\alpha-1}{\alpha}$ of the total cost.

In this paper, we study the special case of the arbitrary-sharing connection game in which there are only two source nodes. We call it the *two-source connection game*. We also consider it a special case in which there are no auxiliary nodes. Notice that the games we consider are similar to the multicast and the broadcast games, except that in our games there are two source nodes rather than one. The main findings of our paper are as follows: In the two-source connection game, we show that there exists a 2-approximate Nash equilibrium that is socially optimal. For the further special case in which there are no auxiliary nodes, we prove that there always exists a $\frac{3}{2}$-approximate Nash equilibrium that is socially optimal and which is computable in polynomial time.

The rest of this paper is organized as follows: Section 2 introduces the problem and presents preliminary results from the literature. Section 3 introduces some terminology and presents the ideas commonly used in showing the existence and computability of approximate Nash equilibrium solutions in arbitrary-sharing network formation games. Section 4 presents our result for the two-source connection game. Section 5 studies the special case of the two-source connection game in which there are no auxiliary nodes. Section 6 summarizes our findings and outlines avenues for future research.

2. Model and preliminaries

In the rest of the paper, we consider a given undirected graph $G = (V, E, c)$. The value $c(e) \geq 0$ denotes the cost of building edge $e \in E$.

Let $N = \{1, 2, \ldots, n\}$ be the set of agents (players). A *connection game* associates each $i \in N$ with a pair (s_i, t_i), where $s_i, t_i \in V$. We imagine that agent i sits at a terminal node t_i and wishes to be connected to a source node s_i. Thus, she wants a path to be built between the nodes s_i and t_i. We use $\mathcal{G}$ to denote a connection game instance. Let $S \subset V$ denote the subset of source nodes in $\mathcal{G}$.

For ease of reference, we may speak of a node t_i as an agent (agent i). But notice that multiple agents may sit at the same terminal node (i.e. for $i, j \in N$, we may have $t_i = t_j$). Furthermore, multiple agents may wish to connect to the same source node. We say that a node is *auxiliary* if it is not a source or a terminal node for some agent.

We assume that the cost-sharing scheme is arbitrary. Therefore, for agent i, her strategy is a *payment vector* $p_i : E \to \mathbb{R}_{\geq 0}$, where $p_i(e)$ denotes how much agent i contributes to the cost of edge e. Agents' strategy profile is a payment scheme $p = (p_1, \ldots, p_n)$. We use the terms "strategy profile" and "payment scheme" interchangeably.

A subgraph G' of G consists of a subset of edges and the associated edge costs and nodes. The cost of the subgraph G' is the sum of the costs of the edges in G'. Let $c(G')$ denote this cost. A subgraph G' is socially-optimal if it is the minimum-cost subgraph of G that satisfies every agent's connectivity requirement.

An edge e is built if $\sum_{i \in N} p_i(e) \geq c(e)$. Let G_p denote the subgraph of G that exclusively contains the edges built under the payment scheme $p = (p_1, \ldots, p_n)$. If G_p does not contain an $s_i - t_i$ path, agent i's incurred cost is infinite. If G_p contains an $s_i - t_i$ path, agent i's incurred cost is the sum of her contributions for the edges, i.e. $\sum_{e \in E} p_i(e)$.

Let p_{-i} denote the $(n-1)$-dimensional vector of strategies played by all agents except i under the payment vector p. Notice that we can write p as (p_i, p_{-i}). A payment scheme p is a *Nash equilibrium* if for each $i \in N$, the subgraph G_p contains an $s_i - t_i$ path, and there does not exist a strategy p'_i such that the subgraph $G_{(p'_i, p_{-i})}$ contains an $s_i - t_i$ path and $\sum_{e\in E} p'_i(e) < \sum_{e\in E} p_i(e)$. That is, p is a Nash equilibrium if no agent has a lower-payment deviation that keeps her connectivity requirement satisfied.

Since cost-sharing is arbitrary, agent i only cares about how much she must contribute to an edge e so that e is built. This amount, denoted by $c'_i(e)$, is called the *modified cost* of e for agent i. For instance, if $c(e) = 10$, and if other agents' combined contributions for e is 8, then $c'_i(e) = 10 - 8 = 2$. Notice that for a connection game instance $\mathcal{G}$, a payment scheme p is a Nash equilibrium if for each $i \in N$, the sum $\sum_{e\in E} p_i(e)$ is equal to the cost of the shortest path between s_i and t_i under the modified costs for agent i.

Observation 1 below states some properties of Nash equilibrium payment schemes.

Observation 1 (Anshelevich et al. [1]) *For a connection game instance $\mathcal{G}$, let the payment scheme p be a Nash equilibrium. Then, the following properties hold:*

(P1) *G_p is a forest.* [4]

(P2) *Agent i pays a positive amount only for the edges along the unique $s_i - t_i$ path in G_p.*

(P3) *For edge e, the total payment is either $c(e)$ or 0.*

For a connection game instance $\mathcal{G}$, the socially-optimal subgraph is the minimum-cost Steiner forest[5] that connects each terminal node to the associated source node. We use T^* to denote the socially-optimal subgraph. Notice that for a multicast game, T^* connects s to all terminal nodes. We present below the seminal existence result for the multicast game.

Theorem 1 (Anshelevich et al. [1]) *There exists a polynomial-time algorithm that satisfies the following specifications: The algorithm takes as input a multicast game instance $\mathcal{G}$ and a Steiner tree T that connects s to all terminal nodes. Then, it returns:*

- *A Nash equilibrium payment scheme p that builds T, or*
- *a Steiner tree T' such that $c(T') < c(T)$ and T' connect s to all terminal nodes.*

We shortly call the algorithm with the properties in Theorem 1 the ADTW algorithm. Notice that since no Steiner tree is cheaper than T^*, if T^* is given as an input, then the ADTW algorithm returns a Nash equilibrium payment scheme that builds T^*. The ADTW algorithm is not directly applicable if there are multiple source nodes. But in network formation games with arbitrary-sharing, the ADTW algorithm is typically used as a subroutine when showing the existence of approximate Nash equilibrium payment schemes that build the socially-optimal subgraph [1, 25, 26].

A payment scheme p is an α*-approximate Nash equilibrium* if for each $i \in N$, G_p contains an $s_i - t_i$ path, and there is no strategy p'_i such that $G_{(p'_i, p_{-i})}$ contains an $s_i - t_i$ path and $\sum_{e\in E} p'_i(e) < \frac{1}{\alpha}\sum_{e\in E} p_i(e)$. That is, at p, agent i does not have a deviation that makes her payment less than $\frac{1}{\alpha}$ times her pre-deviation payment while keeping her connectivity requirement satisfied. Note that this condition is equivalent to saying that the sum $\sum_{e\in E} p_i(e)$ is no more than α times the cost of the shortest path between s_i and t_i under the

[4]An undirected graph G is called a *forest* if any two vertices are connected by at most one path, i.e. G is an acyclic graph. G is called a *tree* if any two vertices are connected by exactly one path, i.e. G is an acyclic connected graph.

[5]Let $G = (V, E, c)$ be an undirected graph with positive edge weights, and let $S_1, S_2, \ldots, S_k$, for $k \geq 1$, be disjoint subsets of V. A *minimum-cost Steiner forest* is a minimum-cost subgraph G' of G in which any two vertices belonging to the same set S_i are connected. Notice that G' is necessarily a forest, and it may or may not be a tree. For the special case where $k = 1$, G' is referred to as *minimum-cost Steiner tree.* Notice that G' is necessarily a tree for $k = 1$.

modified costs for agent i. Also, note that the properties given in Observation 1 need not hold for approximate Nash equilibrium payment schemes.

A *two-source connection game* is a connection game with two source nodes. We denote the source nodes by s_1 and s_2. A two-source connection game instance may not have a Nash equilibrium, as illustrated in Figure 1 (due to [1]). To see this, suppose by contradiction that the payment scheme $p = (p_1, p_2)$ is a Nash equilibrium. Then, G_p may contain only three edges due to **(P1)**. Without loss of generality, let these edges be a, b, c. Due to **(P2)**, agent 1 may only contribute to the costs of a and b, and agent 2 may only contribute to the costs of b and c. So, it must be that $p_1(a) = 1, p_1(c) = 0, p_2(a) = 0, p_2(c) = 1$. Since G_p contains b, we must have $p_1(b) + p_2(b) = 1$. However, neither agent can contribute a positive amount to b due to her alternate strategy under which she only pays 1 for edge d. Therefore, for this game instance, there exists no Nash equilibrium.

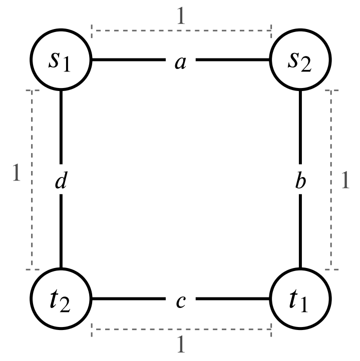

Figure 1. A two-source connection game instance with no Nash equilibrium. The building cost of each edge is 1.

Consider the payment scheme p' where agent 1 builds a, agent 2 builds c, and agents equally share the building cost of b. The cost of p' to each agent is $\frac{3}{2}$. Under p', for agent 1, the modified costs of a, b, c, d are, respectively, $1, \frac{1}{2}, 0, 1$. Similarly, under p', for agent 2, the modified costs of a, b, c, d are, respectively, $0, \frac{1}{2}, 1, 1$. Under the modified costs: For agent 1, the shortest $s_1 - t_1$ path is through d and c, and its total modified cost is 1 and for agent 2, the shortest $s_2 - t_2$ path is through a and d, and its total modified cost is 1. Notice that p' is a $\frac{3}{2}$-approximate Nash equilibrium since for each agent i, $\sum_{e \in E} p_i(e)$ is no more than $\frac{3}{2}$ times the cost of the shortest path between s_i and t_i under the modified costs for i.

Since a two-source connection game instance may not admit a Nash equilibrium, we will concentrate our efforts in proving the existence of a low-factor approximate Nash equilibrium payment scheme. For that, we will make use of the concept of stable payments, which we explain next.

For a connection game instance $\mathcal{G}$, let T be an arbitrary Steiner forest that connects each source node to the terminals that wish to connect it. Assume that agent i plays the strategy p_i, and the remaining agents collectively make the remaining payments necessary to build T. We say that p_i is a *stable payment* for agent i with respect to T if $\sum_{e \in E} p_i(e)$ is equal to the cost of the cheapest $s_i - t_i$ path in G under the modified costs for i. Notice that p is a Nash equilibrium if and only if the payment vector of each agent is a stable payment for her with respect to G_p. Lemma 1 below pertains to the concept of stable payment. It becomes useful when proving the existence of approximate Nash equilibrium results.

Lemma 1 (Anshelevich et al. [1]) *Let p be a payment scheme for a connection game instance $\mathcal{G}$ such that the connectivity requirements of all agents are satisfied in G_p. Assume that the payment vector p_i of each agent*

i can be written as the sum of α payment vectors, $p_i^1, p_i^2, \ldots, p_i^\alpha$, each of which is a stable payment for i with respect to G_p. Then, p is an α-approximate Nash equilibrium payment scheme for $\mathcal{G}$.

The statement in Lemma 1 can be trivially extended to the case where the payment vector of an agent i is a linear combination of a set of stable payments for i with positive coefficients, rather than just a sum. Consider the game instance given in Figure 1. Let $p_1 = (1, \frac{1}{2}, 0, 0)$ and $p_2 = (0, \frac{1}{2}, 1, 0)$. That is, agent 1 builds a, agent 2 builds c, and agents share the building cost of b equally. Let T be the subgraph consisting of the edges a, b, c. Notice that, with respect to T, $(1, 0, 0, 0)$ is a stable payment for agent 1, $(0, 0, 1, 0)$ is a stable payment for agent 2, and $(0, 1, 0, 0)$ is a stable payment for both agents. Then, the payment scheme p is a $\frac{3}{2}$-approximate Nash equilibrium because each agent's payment vector can be written as the sum of a stable payment for her with respect to G_p and a half of another stable payment for her with respect to G_p.

3. Terminology, ideas, and proof techniques

In this section, we present the ideas and techniques introduced in [1] and that are commonly used in the literature when proving the existence and computability of exact or approximate Nash equilibrium solutions in arbitrary-sharing network formation games [25–28]. We define two useful operations on G_p.

Edge contraction: Let p be a payment scheme for a connection game instance $\mathcal{G}$ such that the connectivity requirements of all the agents are satisfied in G_p. Let P be a maximal-length $v_0 - v_k$ path (with at least two edges) in G_p such that for P, each internal node v_i $(0 < i < k)$ is an auxiliary node and the degree of v_i in G_p is 2. The edge contraction operation adds a (v_0, v_k) *compound edge* to G with a building cost of $\sum_{i=0}^{k-1} c((v_i, v_{i+1}))$. The path P in G_p is then replaced with the compound edge (v_0, v_k).

The edge contraction operation is commonly used in arguments when proving the existence of an approximate Nash equilibrium payment scheme that builds the socially-optimal subgraph T^*. The outline of these proofs is as follows: If T^* is a forest, the arguments that we present next can be applied to each tree of T^* separately. Therefore, without loss of generality, assume that T^* is a single tree. Let us apply the edge contraction operation on maximal-length paths of T^* of length at least 2 whose all internal nodes are auxiliary, until T^* does not have any such paths. Note that T^* is now composed of relatively few (simple or compound) edges. Let e be an arbitrary edge of T^*. The removal of e from T^* will divide T^* into two trees, T_1^* and T_2^*. Notice that there is at least one agent, say i, such that $s_i \in T_1^*$ and $t_i \in T_2^*$ or vice versa, since assuming otherwise contradicts T^* being the socially-optimal subgraph. The critical observation here is that building edge e entirely constitutes no more than one stable payment for any agent i with $s_i \in T_1^*$ and $t_i \in T_2^*$ (or vice versa) with respect to T^*. One technique to give a constructive proof of the existence of an α-approximate Nash equilibrium payment scheme that builds T^* is to come up with an assignment of the edges of T^* to agents with the following restrictions. An edge e can be assigned to an agent i only if e is along the unique path between s_i and t_i in T^*, and no agent is assigned more than α edges. To show our existence of approximate Nash equilibrium result, we will use this technique in conjunction with the node contraction idea, which we present next. The node contraction idea enables us to use the ADTW algorithm in the two-source setting.

Node contraction: Without loss of generality, assume that T^* is a tree, since otherwise the arguments that we present next can be applied to each tree of T^* separately. Let T' be the minimal connected subgraph of T^* that spans all the source nodes. The critical observation is the following: Let p be a payment scheme that builds T^*. Let $p_i(T')$ denote the vector of payments by agent i for the edges of T'. Assume that for each agent i, $p_i(T')$ can be written as the sum of k vectors $p_i^1(T'), p_i^2(T'), \ldots, p_i^k(T')$, each of which is a stable payment for i with respect to T^*. That is, no agent i pays more than k stable payments for i with respect to T^* for the edges of T'. Then, there is a $(k+1)$-approximate Nash equilibrium payment scheme $\tilde{p}$ that builds T^*. The payment scheme $\tilde{p}$ can be constructed as follows: For each edge $e \in T'$, and for each agent $i \in N$, we will have $\tilde{p}_i(e) = p_i(e)$. Since the edges of T' are built (by at most k stable payments by each agent with respect to T^*), we can now fix the payments for the edges of T' and contract the nodes of T' into a single node v. Notice that all source nodes (and possibly some terminal nodes) are contracted into v. The connectivity requirements of the agents whose terminals are contracted into v are already satisfied. The remaining edges of T^* are built by the remaining agents so that these agents' connectivity requirements are also satisfied. But note that each of these agents now wishes to connect to v. They can build the remaining edges of T^* by making one stable payment per agent with respect to T^*, and these payments can be decided by the ADTW algorithm.

Structure of the socially-optimal subgraph: The socially-optimal subgraph T^* of a two-source connection game instance $\mathcal{G}$ is composed of a single tree, or two trees such that each contains a source node and the terminal nodes that wish to connect to it. If T^* is composed of two trees, then there exists a Nash equilibrium payment scheme that builds T^*, and one such payment scheme can be computed by running the ADTW algorithm twice (one for each tree). Therefore, in the rest of the text, without loss of generality, we will assume that T^* is composed of a single tree. Assume that we started with T^*, and applied the edge contraction operation on maximal-length paths of T^* of length at least 2 whose all internal nodes are auxiliary, until T^* does not have any such paths. The structure of T^* will then be as depicted in Figure 2. T^* will be composed of an $s_1 - s_2$ path such that each node of this path is the root node of a subtree. Notice that each edge of the $s_1 - s_2$ path is either a simple edge or a compound edge. In Figure 2, the $s_1 - s_2$ path contains k internal nodes named as $v_1, v_2, \ldots,$ and v_k. The subtrees rooted at them are named as $\tau_1, \tau_2, \ldots,$ and τ_k, respectively. For convenience, the subtrees rooted at s_1 and s_2 are named as τ_0 and τ_{k+1}, respectively. Notice that for $1 \le j \le k$, τ_j necessarily contains at least one terminal node, and each leaf node of τ_j is a terminal node. So, if τ_j contains exactly one terminal node, then τ_j itself is composed of the single node v_j. The subtrees τ_0 and τ_{k+1}, on the other hand, may or may not contain terminal nodes. If τ_0 does not contain a terminal node, then τ_0 is composed of s_1. If τ_0 has nodes other than s_1 then each leaf of τ_0 is necessarily a terminal node. Due to symmetry, the arguments for τ_0 also hold for τ_{k+1}, with s_1 replaced with s_2.

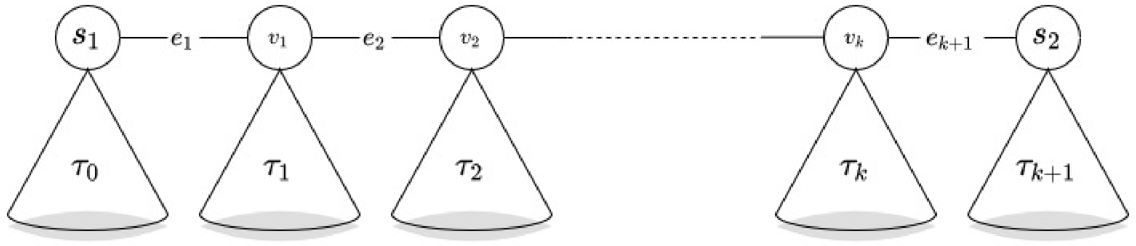

Figure 2. The structure of the socially-optimal subgraph T^* of a two-source connection game instance $\mathcal{G}$.

The above ideas suggest the following for the two-source connection game: Notice that T' is the $s_1 - s_2$ path. If there exists a payment scheme p that builds T^* such that no agent i makes multiple stable payments for i with respect to T^* for the edges of T', then there is a 2-approximate Nash equilibrium payment scheme $\tilde{p}$ that builds T^*. However, for a two-source connection game instance, it may be that a payment scheme p may not exist such that p builds T^* and no agent makes multiple stable payments for her in T^* for the edges of T'. To see that, consider the two-source connection game instance given in Figure 1. A socially-optimal subgraph in that instance consists of the edges b, c, and d. Notice that all the edges of T^* are also in T' and there does not exist a payment scheme p that builds T^* such that no agent i makes multiple stable payments for i with respect to T^* for the edges of T'. We prove the existence of a 2-approximate Nash equilibrium payment scheme that builds T^* in Section 4 by making use of an additional novel observation about the structure of T^*.

4. Existence of 2-approximate Nash equilibrium

This section is devoted to the proof of Theorem 2, which establishes the existence of a 2-approximate Nash equilibrium payment scheme that builds the socially-optimal subgraph.

Theorem 2 *For any two-source connection game instance $\mathcal{G}$, there exists a 2-approximate Nash equilibrium payment scheme p that builds the socially-optimal subgraph T^*.*

As explained in Section 3, without loss of generality, we will assume that T^* is composed of a single tree. We will prove Theorem 2 constructively by presenting a two-phase algorithm. Our algorithm takes as input the two-source connection game instance $\mathcal{G}$, and the socially-optimal subgraph T^* for $\mathcal{G}$. In the preprocessing step, we apply the edge contraction operation on maximal-length paths of T^* of length at least 2 whose all internal nodes are auxiliary, until T^* has no such path. Recall from Section 3 that the structure of T^* will then be as depicted in Figure 2. We will use k to denote the number of internal nodes of the $s_1 - s_2$ path. The internal nodes of the $s_1 - s_2$ path are named as $v_1, v_2, \ldots, v_k$, and the edges of the $s_1 - s_2$ path are named as $e_1, e_2, \ldots, e_{k+1}$, as in Figure 2. The subtrees rooted at the nodes of the $s_1 - s_2$ path are, likewise, named as $\tau_0, \tau_1, \ldots, \tau_{k+1}$. We use S_1 and S_2, in order, to denote the set of terminals that wish to connect to s_1 and s_2.

In the first phase of the algorithm (given below as Algorithm 1), we construct payments on the (simple or compound) edges of the $s_1 - s_2$ path (i.e. T') such that for any agent i, the payment she makes for the edges of T' is no more than a stable payment for her in T^*. As we have demonstrated in Section 3, with the game instance depicted in Figure 1 as an example, there may not exist any payment scheme p that builds T^* such that the total payment of each agent i on the edges of T' is no more than a stable payment for her with respect to T^*. Lemma 2 below shows that Algorithm 1 either succeeds in deciding the payment on all the edges of T', or it succeeds in deciding the payment on all but one of the edges of T'. In the former case, we will show Theorem 2 by using the arguments described in Section 3. In the latter case, we show Theorem 2 by using the properties of T^* given in Lemma 2.

Explanation of Algorithm 1 : Algorithm 1 maintains a queue named Q that is initialized to be empty. Algorithm 1 first loops through the subtrees attached to the internal nodes of the $s_1 - s_2$ path ($\tau_1, \tau_2, \ldots, \tau_k$) in the left-to-right order. Recall that each such subtree contains at least one terminal node. At iteration i, exactly one terminal in τ_i builds an edge of the $s_1 - s_2$ path, i.e. pays the building cost of this (simple or compound) edge. If all the terminals in τ_i wish to connect to s_1, then one of them builds an unbuilt edge of the $s_1 - s_2$ path that is at the left of v_i. Notice that there is only one such edge. This is because exactly i of the edges of the $s_1 - s_2$ path are to the left of v_i, and exactly $i-1$ of them are already built in the previous iterations of the for loop. If all the terminals in τ_i wish to connect to s_2, then one of them builds the edge immediately to the right of v_i, i.e. e_{i+1}. Notice that e_{i+1} has not been built in the previous iterations of the for loop. If τ_i has at least one terminal that wishes to connect to s_1, and at least one terminal that wishes to connect to s_2,

Algorithm 1

```
1: Initialize $Q$ as an empty queue
2: for $i = 1$ to $k$ do
3:     if $S_1 \cap \tau_i \neq \emptyset$ AND $S_2 \cap \tau_i \neq \emptyset$ then
4:         Make $t_l \in S_1 \cap \tau_i$ build the left-most unbuilt edge of the $s_1 - s_2$ path
5:         $Q.enqueue(t_r \in S_2 \cap \tau_i)$
6:     else if $S_1 \cap \tau_i \neq \emptyset$ then
7:         Make $t_l \in S_1 \cap \tau_i$ build the left-most unbuilt edge of the $s_1 - s_2$ path
8:     else
9:         Make $t_r \in S_2 \cap \tau_i$ build $e_{i+1}$
10:    end if
11: end for
12: if $S_2 \cap \tau_0 \neq \emptyset$ then
13:     Make $t_r \in S_2 \cap \tau_0$ build the unique unbuilt edge of the $s_1 - s_2$ path
14: else if $S_1 \cap \tau_{k+1} \neq \emptyset$ then
15:     Make $t_l \in S_1 \cap \tau_{k+1}$ build the unique unbuilt edge of the $s_1 - s_2$ path
16: else if $Q$ is not empty then
17:     Let $t_r \leftarrow Q.dequeue()$
18:     Make $t_r$ build the unique unbuilt edge of the $s_1 - s_2$ path
19: end if
```

then one of the terminals that wish to connect to s_1 builds an unbuilt edge of the $s_1 - s_2$ path that is at the left of v_i, and one of the terminals that wish to connect to s_2 is enqueued to Q for payment afterward.

At the termination of the for loop, exactly k agents built a distinct edge of the $s_1 - s_2$ path that is along the unique path between her terminal and source nodes in T^*. Therefore, all such agents made at most one stable payment with respect to T^* so far. Let e_l be the unique edge of the $s_1 - s_2$ path that is not built upon the termination of the for loop. Notice that if there exists a terminal node t_r in τ_0 that wishes to connect to s_2, then t_r can built e_l with a payment of no more than one stable payment for her with respect to T^* since e_l is on the unique $t_r - s_2$ path in T^*. Similarly, e_l can be built by a terminal node t_l in τ_{k+1} that wishes to connect to s_1 with a payment of no more than one stable payment for her with respect to T^*. We next show that any terminal node t_r enqueued to Q can also build e_l with a payment of no more than one stable payment for her with respect to T^*. Let $t_r \in \tau_i$ be a terminal node that is enqueued to Q at iteration i of the for loop. Note that exactly i of the edges of the $s_1 - s_2$ path are to the left of v_i, and in the first $i-1$ iterations of the for loop exactly $i-1$ of those edges are built. At iteration i, a terminal $t_l \in \tau_i(t_l \neq t_r)$ builds the remaining edge of the $s_1 - s_2$ path that is at the left of v_i. Therefore, all the edges of the $s_1 - s_2$ path that are to the left of v_i are already built at the end of iteration i. Since e_l is not built upon the termination of the for loop, it must be the case that e_l is to the right of v_i. But then, e_l is along the unique $t_r - s_2$ path in T^*, and t_r can build e_l with one stable payment for her with respect to T^*. The if-else statement below the for loop conditions on these three cases. Therefore, if e_l is not built at the termination of Algorithm 1, it must be the case that all the terminals in τ_0 wish to connect to s_1, all the terminals in τ_{k+1} wish to connect to s_2, and for $1 \leq j \leq k$, all the terminals in τ_j wish to connect to the same source.

Our final remark in the analysis of Algorithm 1 is about the terminals in a subtree that is to the right of e_l. Let $l \leq j \leq k$. Exactly j edges of the $s_1 - s_2$ path are to the left of v_j, and exactly $j-1$ of those edges are built in the first $j-1$ iteration of the for loop. If there was a terminal in τ_j that wishes to connect to s_1, the left-most unbuilt edge, i.e. e_l, would be built at iteration j of the for loop. Since e_l is not built upon the termination of the for loop, all the terminals in τ_j wish to connect to s_2. All the findings obtained in the analysis of Algorithm 1 are banded together in Lemma 2 below.

Lemma 2 *For each agent i, the payment assigned to her for the edges of T' by Algorithm 1 is no more than one stable payment for i with respect to T^*. Moreover, when Algorithm 1 terminates either $k+1$ or k edges of T' are built. Assume exactly k edges of T' are built upon the termination of Algorithm 1, and let e_l denote the edge of T' that is not built. Then, all of the following statements are true.*
(S1) *All the terminals in τ_0 wish to connect to s_1.*
(S2) *For $0 \leq j \leq k+1$, all the terminals in τ_j wish to connect to the same source.*
(S3) *For $j \geq l$, all the terminals in τ_j wish to connect to s_2.*

The second phase of our algorithm consists of contracting nodes of the input graph and running the ADTW algorithm for a subset of players on the contracted graph. The proof relies on Lemma 2 and the following fact. For any subtree of T^* rooted at a node of the $s_1 - s_2$ path, the minimum-cost Steiner tree that connects all the terminals in this subtree, and the root of the subtree, is the subtree itself. This statement is true since otherwise, replacing this subtree with the minimum-cost Steiner tree of the aforementioned nodes in T^* would give a tree cheaper than T^* that connects all the terminals and the source nodes, which will contradict T^* being the socially-optimal subgraph.

We first consider the case, where all $k+1$ edges of T' are built upon the termination of Algorithm 1. In this case, we fix the payments for the edges of T' and contract the nodes of T' into a single node v. Recall that both source nodes and possibly some terminal nodes are now contracted into v. The connectivity requirements of the agents whose terminals are contracted into v are already satisfied. However, all the agents whose terminals are not contracted into v now wish to connect to v. So, at the second phase of the algorithm, all we need is to find a Nash equilibrium payment scheme for the resulting multicast game and this can be done by running the ADTW algorithm since the edges of $T^* \setminus T'$ is a socially-optimal subgraph for the resulting multicast game.

We now consider the case, where exactly k edges of T' are built upon the termination of Algorithm 1. We fix the payments for the edges of the $v_l - s_2$ path and contract the nodes $v_l, v_{l+1}, \ldots, v_k$, and s_2 into a single node $\tilde{s_2}$. We next find a Nash equilibrium payment scheme for the multicast game played by the agents, whose terminals reside in a subtree rooted at a contracted node. Recall from Lemma 2 that all those agents wish to connect to $\tilde{s_2}$, and the union of the subtrees $\tau_l, \tau_{l+1}, \ldots, \tau_{k+1}$ is a socially-optimal subgraph of this multicast game. Therefore, a Nash equilibrium payment scheme for this multicast game can be obtained by running the ADTW algorithm. Since the payments on the edges of the subtrees rooted at some node contracted into $\tilde{s_2}$ are now decided, we fix the payments on those edges and contract all these subtrees into $\tilde{s_2}$ as well.

We next fix the payments for the edges of the $s_1 - v_{l-1}$ path and contract the nodes $s_1, v_1, \ldots, v_{l-1}$, into a single node $\tilde{s_1}$. Recall that for a subtree attached to $\tilde{s_1}$, either all the terminals in this subtree wish to connect to $\tilde{s_1}$, or all the terminals in this subtree wish to connect to $\tilde{s_2}$. We next find a Nash equilibrium payment scheme for the multicast game played by the agents that wish to connect to $\tilde{s_1}$. Since the union of the subtrees those agents reside is a socially-optimal subgraph for this multicast game, this can be done by running the ADTW algorithm for this multicast game. We then contract these subtrees into $\tilde{s_1}$. Notice that T^*, at this point, is composed of the subtrees rooted at $\tilde{s_1}$ (all the terminals in all such subtrees wish to connect to $\tilde{s_2}$), and the edge e_l. Since the resulting game is also a multicast game, and the remaining edges of T^* is a socially-optimal subgraph of this multicast game, a Nash equilibrium payment scheme for this multicast game can be decided by the ADTW algorithm. This completes the proof of Theorem 2.

5. Special case: Each nonsource node is a terminal node

This section is devoted to the special case of the two-source connection game, where each nonsource node is necessarily a terminal node of some agent. For this special case, Theorem 3 below states the existence and polynomial-time computability of a $\frac{3}{2}$-approximate Nash equilibrium payment scheme that builds T^*.

Theorem 3 *Let $\mathcal{G}$ be a two-source connection game instance such that there are no auxiliary nodes in the input graph G. A $\frac{3}{2}$-approximate Nash equilibrium payment scheme that builds the socially-optimal subgraph can be computed in polynomial time.*

Proof We first prove that T^* can be computed in polynomial time. Let S_1 and S_2 denote the set of terminals that wish to connect to s_1 and s_2, respectively. Recall that T^* is either a single tree, or it is composed of two trees. If T^* is a single tree, then it is a minimum spanning tree of the input graph G, which can be computed in polynomial time. If T^* is composed of two trees (say T_1^* and T_2^*), then the vertex set of T_1^* is composed of s_1 and S_1, the vertex set of T_2^* is composed of s_2 and S_2, and T^* is the minimum-cost such a forest. The following algorithm finds T^*.

The algorithm first computes the minimum spanning tree T of G in polynomial time and stores it. It then computes subgraphs $G_1 = (V_1, E_1, c_1)$ and $G_2 = (V_2, E_2, c_2)$ of $G = (V, E, c)$ that are defined as follows:

- $V_1 = \{s_1\} \cup S_1$, and $V_2 = \{s_2\} \cup S_2$,
- $E_1 = \{e = (i, j) \in E : i, j \in V_1\}$, and $E_2 = \{e = (i, j) \in E : i, j \in V_2\}$.

That is, E_1 consists of the edges with two endpoints in V_1. Similarly, E_2 consists of the edges with two endpoints in V_2. The building costs of the edges are not changed. Notice that it may be the case that some edges of G are neither in G_1 nor in G_2. Observe that $V_1 \cap V_2 = \emptyset$ if and only if all the agents sitting at the same terminal node wish to connect to the same source. If $V_1 \cap V_2 \neq \emptyset$, then the algorithm returns T. If at least one of G_1 and G_2 is not connected, then the algorithm returns T. If $V_1 \cap V_2 = \emptyset$, and both G_1 and G_2 are connected, the algorithm computes the minimum spanning trees T_1 and T_2 of G_1 and G_2, respectively. The algorithm returns $T_1 \cup T_2$ if $c(T_1 \cup T_2) \leq c(T)$, and it returns T otherwise. This algorithm runs in polynomial time since it is composed of three calls to a minimum spanning tree algorithm (such as Prim's algorithm, or Kruskal's algorithm), and the construction of two subgraphs of the given input graph. The correctness of the algorithm follows from the fact that it exhaustively considers all possible cases.

We now prove the existence of a $\frac{3}{2}$-approximate Nash equilibrium payment scheme that builds T^*. If T^* is composed of two trees T_1^* and T_2^*, then T_1^* is the socially-optimal subgraph for the multicast game of the agents S_1, and T_2^* is the socially-optimal subgraph for the multicast game of the agents S_2. Therefore, a Nash equilibrium payment scheme that builds T^* can be computed by running the ADTW algorithm twice (once for each tree). Thus, without loss of generality, we will assume that T^* is a tree. The structure of T^* is as depicted in Figure 2, and additionally, all the edges of T^* are now simple edges. We prove the theorem constructively by presenting an algorithm that constructs a payment scheme that builds T^*.

Our algorithm first determines the payments for the edges of $T^* \setminus T'$, i.e. the edges in the subtrees $\tau_0, \tau_1, \ldots, \tau_{k+1}$. One agent from each nonroot terminal builds the edge between her terminal node and the parent of her terminal node. Notice that each agent builds an edge along the unique path between her terminal and source node pair in T^*, and thus, the payment assigned to each such agent constitutes no more than one stable payment for her in T^*.

Next, we determine the payments on k of the edges of T', i.e. the $s_1 - s_2$ path. One agent from each internal node of the $s_1 - s_2$ path $(v_1, v_2, \ldots, v_k)$ builds a single edge of the $s_1 - s_2$ path. Assignment of the edges to the agents is done in a way similar to what is done in the for loop of Algorithm 1. That is, we loop through these terminals in left-to-right order and process terminal v_i at iteration i of the loop. The processing of v_i is done as in the following. If v_i is the terminal node of an agent that wishes to connect to s_1, then that agent builds the left-most unbuilt edge of the $s_1 - s_2$ path. Otherwise, v_i is the terminal node of an agent that wishes to connect to s_2, and that agent builds e_{i+1}. Since agents build edges along the unique path between their terminal and source node pairs in T^*, the payment assigned to each agent at that stage of the algorithm constitutes no more than one stable payment for her in T^*.

At that point of the algorithm, all the edges of T^*, except one edge of the $s_1 - s_2$ path, are built, and no agent made more payment than one stable payment for her in T^*. Let e_l be the edge that is not built yet. We say that an agent i *witnesses* an edge e of T^* if e is along the unique path between s_i and t_i in T^*. Let r be the number of agents witnessing e_l. Notice that $r \neq 0$ since otherwise, removal of e_l from T^* creates a cheaper subgraph that satisfies the connectivity requirements of all the agents, which contradicts T^* being the socially-optimal subgraph. If $r \geq 2$, the agents witnessing e_l will share its cost equally, i.e. each witnessing agent will pay an amount of $\frac{c(e_l)}{r}$ toward the construction of e_l. This payment constitutes no more than $\frac{1}{r}$ stable payment for each witnessing agent in T^*, and thus, the payment scheme given is a $(1 + \frac{1}{r})$-approximate Nash equilibrium, where $1 + \frac{1}{r} \leq \frac{3}{2}$ as desired.

We now consider the case $r = 1$. Let agent i be the only agent that witnesses e_l. Let edge e_o be the edge that agent i built in the previous stages of the algorithm. We make agent i build e_l instead of e_o. Notice that this payment constitutes a stable payment for agent i in T^* since e_l is also along the unique path between s_i and t_i in T^*. Let r' denote the number of agents witnessing e_o. If $r' \geq 2$, then the payment scheme where the agents witnessing e_o share the cost of e_o equally is a $(1 + \frac{1}{r'})$-approximate Nash equilibrium as we described above. If $r' = 1$ we make agent i build both e_o and e_l. All we need is to show that this payment constitutes a stable payment for agent i in T^*. Notice that the connectivity requirements of all the agents, except i, are satisfied in $T^* \setminus \{e_o, e_l\}$. Let agent i play her best response strategy to the strategies of the other agents, i.e. build the path between s_i and t_i with the lowest modified cost. If the modified cost of this path is cheaper than $c(e_o) + c(e_l)$, this will contradict with T^* being the socially-optimal subgraph. Therefore, the best response of agent i to the strategies of the other agents is to build both e_o and e_l. And thus, building both e_o and e_l is a stable payment for agent i in T^*. Notice that the payment scheme constructed in the case $r' = 1$ is an exact Nash equilibrium. This completes the proof.

□

6. Conclusion and Future Research Directions

We studied the arbitrary-sharing connection game under the restriction that there are exactly two source nodes. We proved the existence of a 2-approximate Nash equilibrium payment scheme that builds the socially-optimal subgraph. In the further restricted setting where there are no auxiliary nodes, we proved the polynomial-time computability of a $\frac{3}{2}$-approximate Nash equilibrium payment scheme that builds the socially-optimal subgraph. One evident direction for future research is investigating the existence of approximate Nash equilibrium payment schemes with a lower approximation ratio than shown in this paper.

Another direction for future research, one that may be quite interesting, is considering restrictions on the strategy space for the arbitrary-sharing network formation games. For these games, since the strategy space is extensive, it is rare to find results in the literature proving the nonexistence of an α-approximate Nash equilibrium for some α. But suppose that attention is confined to the payment schemes that satisfy **(P2)**, i.e. when agent, say i, makes positive contributions only along a simple $s_i - t_i$ path. The graph in Figure 1 has no auxiliary nodes and the payment scheme constructed by the algorithm given in Section 5 necessarily satisfies **(P2)**. For the game instance depicted in Figure 1, it is easy to see that no payment scheme that satisfies **(P2)** is an α-approximate Nash equilibrium solution for any $\alpha < \frac{3}{2}$. Thus, the result in Theorem 3 is tight under the additional restriction that payment schemes satisfy **(P2)**. But now consider the payment scheme for which the payments of agents 1 and 2 for a, b, c, d are $(\frac{1}{2}, \frac{3}{4}, 0, \frac{1}{4})$ and $(\frac{1}{2}, \frac{1}{4}, 0, \frac{3}{4})$, in order. It can be verified that this payment scheme is a $\frac{6}{5}$-approximate Nash equilibrium, and hence, in a sense, it is closer to a Nash equilibrium. But under this payment scheme, agents are contributing to the costs of the edges that they are not using, and such behavior is not expected from strategic agents. Therefore, future research may consider restricting payment schemes by the condition **(P2)**. From a technical point of view, as exemplified above, this restriction will make it easier to prove tight existence of approximate Nash equilibrium results.

Acknowledgment

All authors contributed to the proof of the results and the preparation of the manuscript.

References

[1] Anshelevich E, Dasgupta A, Tardos E, Wexler T. Near-optimal network design with selfish agents. Theory of Computing 2008; 4 (1): 77-109. doi: 10.4086/toc.2008.v004a004

[2] Russell SJ, Norvig P. Artificial intelligence - a modern approach: the intelligent agent book. Upper Saddle River, New Jersey, USA: Prentice Hall, 1995.

[3] Elzayn H, Colini-Baldeschi R, Lan B, Schrijvers O. Equilibria in Auctions with Ad Types. In: ACM Web Conference (WWW); Lyon, France; 2022. pp. 68-78.

[4] Ciocan DF, Iyer K. Tractable Equilibria in Sponsored Search with Endogenous Budgets. Operations Research 2021; 69 (1): 227-244. doi: 10.1287/opre.2020.2052

[5] Küpçü A, Safavi-Naini R. Smart Contracts for Incentivized Outsourcing of Computation. In: 5th International Workshop on Cryptocurrencies and Blockchain Technology (CBT); Darmstadt, Germany; 2021. pp. 245-261.

[6] Li T, Zhao Y, Zhu Q. The role of information structures in game-theoretic multi-agent learning. Annual Reviews in Control 2022; 53: 296-314. doi: 10.1016/j.arcontrol.2022.03.003

[7] Nisan N, Roughgarden T, Tardos E, Vazirani VV. Algorithmic Game Theory. Cambridge, UK: Cambridge University Press, 2007.

[8] Nash JF. Equilibrium points in n-person games. Proceedings of the National Academy of Sciences (PNAS) 1950; 36: 48-49.

[9] Koutsoupias E, Papadimitriou CH. Worst-case equilibria. In: 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS); Trier, Germany; 1999. pp. 404-413.

[10] Anshelevich E, Dasgupta A, Kleinberg J, Tardos E, Wexler T et al. The price of stability for network design with fair cost allocation. SIAM Journal on Computing 2008; 38 (4): 1602-1623. doi: 10.1137/070680096

[11] Tardos E. Network Games. In: 36th Annual ACM Symposium on Theory of Computing (STOC); Chicago, IL, USA; 2004. pp. 341-342.

[12] Tardos E, Wexler T. Network formation games and the potential function method. In: Nisan N, Roughgarden T, Tardos E, Vazirani VV (editors). Algorithmic Game Theory. 1st ed. Cambridge, UK: Cambridge University Press, 2007, pp: 487-516.

[13] Myerson RB. Graphs and cooperative games. Mathematics of Operations Research 1977; 2 (3): 225-229. doi: 10.1287/moor.2.3.225

[14] Jackson MO. A survey of network formation models: stability and efficiency. In: Demange G, Wooders M (editors). Group formation economics: networks, clubs and coalitions. 1st ed. Cambridge, UK: Cambridge University Press, 2005, pp. 11-57.

[15] Fabrikant A, Luthra A, Maneva EN, Papadimitriou CH, Shenker S. On a network creation game. In: 22nd ACM Symposium on Principles of Distributed Computing (PODC); Boston, MA, USA; 2003. pp. 347-351.

[16] Johari R, Mannor S, Tsitsiklis JN. A contract-based model for directed network formation. Games and Economic Behavior 2006; 56 (2): 201-224. doi: 10.1016/j.geb.2005.08.010

[17] Anshelevich E, Shepherd FB, Wilfong GT. Strategic network formation through peering and service agreements. Games and Economic Behavior 2011; 73 (1): 17-38. doi: 10.1016/j.geb.2011.01.002

[18] Rosenthal RW. A class of games possessing pure-strategy Nash equilibria. International Journal of Game Theory 1973; 2 (1): 65-67. doi: 10.1007/BF01737559

[19] Moulin H, Shenker S. Strategyproof sharing of submodular costs: budget balance versus efficiency. Economic Theory 2001; 18: 511-533. doi: 10.1007/PL00004200

[20] Herzog S, Shenker S, Estrin D. Sharing the cost of multicast trees: an axiomatic analysis. IEEE/ACM Transactions on Networking 1997; 5 (6): 847-860. doi: 10.1109/90.650144

[21] Feigenbaum J, Papadimitriou CH, Shenker S. Sharing the cost of multicast transmissions. Journal of Computer and System Sciences 2001; 63 (1): 21-41. doi: 10.1006/jcss.2001.1754

[22] Bilò V, Caragiannis I, Fanelli A, Monaco G. Improved lower bounds on the price of stability of undirected network design games. Theory of Computing Systems 2013; 52 (4): 668-686. doi: 10.1007/s00224-012-9411-6

[23] Li J. An $O(\frac{\lg n}{\lg \lg n})$ upper bound on the price of stability for undirected Shapley network design games. Information Processing Letters 2009; 109 (15): 876-878. doi: 10.1016/j.ipl.2009.04.015

[24] Bilò V, Flammini M, Moscardelli L. The price of stability for undirected broadcast network design with fair cost allocation is constant. Games and Economic Behavior 2020; 123: 359-376. doi: 10.1016/j.geb.2014.09.010

[25] Anshelevich E, Caskurlu B. Exact and approximate equilibria for optimal group network formation. Theoretical Computer Science 2011; 412 (39): 5298-5314. doi: 10.1016/j.tcs.2011.05.049

[26] Anshelevich E, Caskurlu B. Price of stability in survivable network design. Theory of Computing Systems 2011; 49 (1): 98-138. doi: 10.1007/s00224-011-9317-8

[27] Anshelevich E, Caskurlu B, Hate A. Strategic multiway cut and multicut games. Theory of Computing Systems 2013; 52 (2): 200-220. doi: 10.1007/s00224-011-9380-1

[28] Anshelevich E, Caskurlu B, Kar K, Zhang H. Capacity allocation games for network-coded multicast streaming. IEEE/ACM Transactions on Networking 2014; 22 (2): 595-607. doi: 10.1109/TNET.2013.2255890

[29] Harks T, Hoefer M, Schedel A, Surek M. Efficient black-box reductions for separable cost sharing. Mathematics of Operations Research 2021; 46 (1): 134-158. doi: 10.1287/moor.2020.1050