Non-cooperative Tree Creation
(Extended Abstract)

Martin Hoefer [*]

Department of Computer & Information Science,
Konstanz University, Box D 67, 78457 Konstanz, Germany
hoefer@inf.uni-konstanz.de

Abstract. In this paper we consider the *connection game*, a simple network design game with independent selfish agents that was introduced by Anshelevich et al [4]. In addition we present a generalization called *backbone game* to model hierarchical network and backbone link creation between existing network structures. In contrast to the connection game each player considers a number of groups of terminals and wants to connect at least one terminal from each group into a network. In both games we focus on an important subclass of *tree games*, in which every feasible network is guaranteed to be connected.

For tree connection games, in which every player holds 2 terminals, we show that there is a Nash equilibrium as cheap as the optimum network. We give a polynomial time algorithm to find a cheap $(2 + \epsilon)$-approximate Nash equilibrium. It can be generalized to find a cheap $(3.1 + \epsilon)$-approximate Nash equilibrium for tree connection games with any number of terminals per player in polynomial time. This improves the guarantee of the only previous algorithm for the problem [4], which returns a $(4.65+\epsilon)$-approximate Nash equilibrium. Tightness results for the analysis of all algorithms are derived.

For single source backbone games, in which each player wants to connect one group to a common source, there is a Nash equilibrium as cheap as the optimum network and a polynomial time algorithm to find a cheap $(1 + \epsilon)$-approximate Nash equilibrium.

1 Introduction

Analyzing networks like the Internet, which is created and maintained by independent selfish agents with relatively limited goals, has become a research area attracting a lot of interest. In particular, there have been many approaches to characterize computational networking aspects using game-theoretic considerations. Naturally, in such games the existence, cost and computation of *stable* solutions are most important. Stable networks are not necessarily cheap or optimized, yet in many situations a central institution interested in optimizing social desiderata has some means of controlling agent behavior. In these cases it is important to understand the dynamics in influencing agents and to explore the boundary between stability and social welfare. Hence, it is of interest to characterize the price of stability [3], which is the ratio of the cost of the best Nash equilibrium over the cost of a socially optimum solution. This captures *how good stability can get*. Recently this measure has been studied in routing and network creation games [3, 4, 14, 20]. The more prominent measure is the price of anarchy [16] describing the cost of the worst instead of the best Nash equilibrium. It has received attention in networking problems, for instance routing [19],

[*] Supported by DFG Research Training Group 1042 "Explorative Analysis and Visualization of Large Information Spaces"

facility location [21] and load balancing [8, 16].
An important aspect of networks is their topology. In this paper we consider and extend the *connection game*, a game-theoretic model for network topology creation introduced by Anshelevich et al [4]. In a connection game each of the k selfish agents has a connectivity requirement, i.e. she holds a number of terminals at some nodes in a given graph and wants to connect these nodes into a component. Possible edges have costs, and agents offer money to purchase them. Once the sum of all agents offers for an edge exceeds its cost, it is considered *bought*. Bought edges can be used by *all* agents to establish their connection, no matter whether they contribute to the cost. Each agent tries to fulfill her connection requirement at the least possible cost.
In the connection game it might be optimal for the agents to create disconnected local subnetworks. The Internet, however, receives its power as a platform for information sharing and electronic trade from the fact that it is *globally* connected - and not only a collection of disconnected, distributed subnetworks. Hence, it is reasonable to assume that agents to some extent have an interest in being connected to the network of other agents. We incorporate this idea by focusing on *tree connection games* - connection games, in which every feasible solution is connected. Furthermore we study the interest in globally connected networks in an extended model, which we call the *backbone game*. It serves to analyze the creation of hierarchical networks. We assume a scenario with existing, globally unconnected subnetworks of small capacity. Each agent wants to connect a set of subnetworks with a connected network of high performance backbone links. Backbone links can start and end at any terminal in the subnetworks, so we can consider subnetworks as groups of terminals in the graph and adjust the connectivity requirements to be present between certain groups. Each player must connect at least one terminal of each of her groups into a connected network at the least cost. Purchasing and using edges works similar to the connection game.

Related Work The connection game was introduced and studied in [4], where a variety of results were presented. Both prices of anarchy and stability are $\Theta(k)$. It is NP-complete to determine, whether a given game has a Nash equilibrium at all. A polynomial time algorithm was presented that finds a $(4.65 + \epsilon)$-approximate Nash equilibrium on a 2-approximate network. For the single-source case, in which each player needs to connect a single terminal to a common source, a polynomial time algorithm was given that finds a $(1 + \epsilon)$-approximate Nash equilibrium on a 1.55-approximate solution. We denote these algorithms by ADTW-SS for the single source and ADTW for the general case. In [3] the authors used adjusted connection games to study the performance of the Shapley value cost sharing protocol. Each edge is bought in equal shares by each player using it to connect its terminals. The price of stability in this game is $O(\log k)$. Furthermore extended results were presented, e.g. on delays, weighted games and best-response dynamics. Recently, connection games have been studied in a geometric setting. In [14] bounds were shown on the price of anarchy and the minimum incentives to deviate from an assignment purchasing the socially optimum network. The case of 2 players and 2 terminals per player was characterized in terms of prices of anarchy and stability, approximate equilibria and best-response dynamics.

A network creation game of different type was considered in [2,7,9]. Here each agent corresponds to a node and can only create edges that are incident to her node. Similar settings are recently receiving increased attention in the area of social network analysis [5, 13]. An overview over recent developments in the area of social network design games is given e.g. in [15]. In the context of large-scale computational networks, however, a lot of these models lack properties like arbitrary cost sharing of edges and complex connectivity requirements.

Our Results In this paper we will consider tree connection games (TCG) and single source backbone games (SBG). The games exhibit connection requirements such that every feasible solution network is connected. Both are different generalizations of single source games studied in [4], and we analyze them in a similar fashion with respect to strict and approximate deterministic pure-strategy Nash equilibria. We are especially interested in polynomial time algorithms for a two-parameter optimization problem: Try to assign payments to the players such that the purchased feasible network is cheap and the incentives to deviate are low. In Section 2 we show that for any path tree connection game - a TCG with two terminals per player - the price of stability is 1. We outline an algorithm that allocates edge costs of a centralized optimum solution to players such that no player has an incentive to deviate. As this algorithm is not efficient, we show in Section 3 how to find a 2-approximate Nash equilibrium purchasing the optimum network for TCGs with any number of terminals per player. It can be translated into a polynomial time algorithm for $(3.1+\epsilon)$-approximate Nash equilibria purchasing a network of cost at most 1.55 times the optimum network cost. This improves over ADTW that provides $(4.65+\epsilon)$-approximate Nash equilibria.
In addition we derive a tightness argument for the design technique of our algorithm and ADTW. Both algorithms consider only the optimum network and to use a bounding argument for deriving approximate Nash equilibria. We show that both are optimal with respect to the class of deterministic algorithms working only on the optimum network. Hence, methods with better performance guarantees can still be found, however, they must explicitly incorporate the cost and structure of possible deviations. This significantly complicates their design and analysis.
In Section 4 we introduce the backbone game. Some results from the connection game translate directly: (1) both prices of anarchy and stability are in $\Theta(k)$, (2) it is NP-complete to determine, whether a given game has a Nash equilibrium and (3) there is a lower bound of $\left(\frac{3}{2}-\epsilon\right)$ on approximate Nash equilibria purchasing the optimum network. Here we show that for SBGs the price of stability is 1. A $(1+\epsilon)$-approximate Nash equilibrium can be found in polynomial time. We outline three extensions, for which the procedure delivers the same results: (1) games with a single source group, (2) games with a directed graph, in which players need a direct connection to the single source and (3) games, in which each player i has a threshold max(i) and would like to stay unconnected if the assigned cost exceeds max(i).
All our results extend to games, in which the centralized optimum forest is composed of trees representing TCGs or SBGs, resp. It is, however, NP-hard to decide, whether a connection or backbone game has such a property.

2 The Price of Stability

Connection Games The connection game for k players is defined as follows. For each game there is an undirected graph $G = (V, E)$, and a nonnegative cost $c(e)$ associated with each edge $e \in E$. Each player owns a set of terminals located at nodes of the graph that she wants to connect. A strategy for a player i is a function p_i, which specifies for each edge the amount $p_i(e)$ that i offers for the purchase of e. If the sum of the offers of all players to an edge e exceeds $c(e)$, the edge is bought. Bought edges can be used by all players to connect their terminals, no matter whether they contribute to the edge costs or not. An (a-approximate) Nash equilibrium is a payment scheme such that no player can reduce $\sum_{e \in E} p_i(e)$ (by more than a factor of a) by unilaterally choosing a different strategy. Note that each player insists on connecting her terminals, and hence considers only such strategies as alternatives.
The problem of finding an optimum centralized network for all players and an optimum strategy for a single player are the classic network design problems of the Steiner network [1, 12] and the Steiner tree [18], respectively. For the rest of this paper we will denote an optimum centralized network by T^*. The subtree of T^* that player i uses to connect her terminals is denoted by T^i.

Tree Connection Games We will deal with an interesting class of connection games, the tree connection games (TCG). TCGs are connection games with tree connection requirements.

Definition 1. *In a connection game there are tree connection requirements if for any two nodes v_1 and v_{l+1} carrying terminals, there is a sequence of players $i_1, \dots, i_l$ and nodes $v_2, \dots v_l$ such that player i_j has terminals at nodes v_j and v_{j+1} for $j = 1, \dots, l$.*

Note that if a single player h would own all terminals of the k players in a TCG, her optimum network would be the same as in the distributed case. Hence, with tree connection requirements the k players imitate the behavior of a global player with respect to feasible solutions. A TCG can be cast as a splitting of a single global player into k players, which preserves the connection requirements.
For the subclass of TCG with 2 terminals per player we will use the term path tree connection game (PTCG). A first observation is that the price of anarchy of the PTCG is k. This is straightforward with an instance consisting of two nodes and two parallel edges, where each node holds a terminal of each player. One edge e_1 has cost k, the other edge e_2 a cost of 1. If each player is assigned to purchase a share of 1 of e_1, the solution forms a Nash equilibrium.
Throughout the paper we will use an elimination order of players, who get assigned payments and are removed. As candidates for elimination we consider leaf players.

Definition 2. *A player owns a* lonely terminal t*, if t is located at a node, where no terminal of another player is located. A player i in a TCG is a* leaf player*, if she owns a lonely terminal, and there is at most one node with a non-lonely terminal of i.*

Our general algorithmic framework for deriving exact and approximate Nash equilibria is as follows. In each iteration we pick a player, assign payments, remove the player and reduce the edge costs by the amount she paid. The algorithm continues until no player is left. In the following outline we assume $c^1(e) = c(e)$ for all $e \in E$.

Algorithmic Framework

```
1. For $iter \leftarrow 1$ to $k$
2.    $i$ is a leaf player if possible; otherwise an arbitrary player
3.    Determine $p_i$ using $c^{iter}$
4.    Set $c^{iter+1}(e) \leftarrow c^{iter}(e) - p_i(e)$ for all $e \in E$
5.    Remove $i$, contract edges of cost 0
```

Theorem 1. *The price of stability in the PTCG is 1.*

Proof. To prove our theorem we need the following technical lemma. Consider a game in which T^* contains all nodes from the graph G. There are two players h and i and a single source terminal at a node s shared by both players. Player i holds exactly one additional terminal.

Lemma 1. *In the described game there is a Nash equilibrium as cheap as T^*.*

Algorithm 1

```
1. For each edge $e$ in reverse BFS order
2.    Find cheapest deviations $A_h$ and $A_i$ for players $h$ and $i$ under $c'$ and
      given $p_h$ and $p_i$ on $T_e$.
3.    Assign $p_i(e) = \min(c(e),\ c'(A_i) - p_i(T_e))$.
4.    Assign $p_h(e) = \min(c(e) - p_i(e),\ c'(A_h) - p_h(T_e))$.
```

Proof. Algorithm 1 is used to construct a Nash equilibrium purchasing T^* for a game as described in the lemma. It considers edges in reverse BFS order from s with adjusted edge costs. Let T_e denote the part of T^* below an edge e and T_u the part below a node u, where $e \notin T_e$ but $u \in T_u$. When assigning the cost of e we use a cost function c' with $c'(e') = 0$ for $e' \in T^* \backslash T_e$ and $c'(e') = c(e')$ otherwise. A_i and A_h are the cheapest feasible deviation trees excluding e for players i and h, resp. We first focus on the question, whether p_h allows a cheaper deviation for h. In opposite to player i it is not trivial for player h to assume that all edges outside of T_e have cost 0. Consider a node u where multiple subtrees join. We know for each edge $e_1, e_2, e_3, \ldots$ below u that the tree $T_{e_j} + e_j$ is the optimum forest to connect the terminals of T_{e_j} to $T^* \backslash T_{e_j}$. But player h owns terminals in possibly *all* subtrees T_{e_j}. Is there a cheaper forest for h to connect her terminals in T_u to $T^* \backslash T_u$ than her calculated contribution?

Lemma 2. *The payment function p_h constructed by Algorithm 1 allows no cheaper deviation for player h.*

Proof. Let the edges $e_1, \ldots, e_l$ be the edges directly below a node u in T_u. Assume the algorithm was able to assign payments that cover the costs of each $T_{e_j} + e_j$, and that u is the first node, at which $p_h(T_u)$ is not optimal for h.
We create a new cost function c'_h with $c'_h(e') = 0$ for $e' \in T^* \backslash T_u$ and $c'_h(e') = c(e') - p_i(e')$ otherwise. We will see that T^* is the optimum network under c'_h. Suppose the cheapest deviation tree A_h is cheaper than T^* under c'_h (i.e. the contribution of h to T^*). W.l.o.g. A_h includes all edges of cost 0, especially all edges purchased

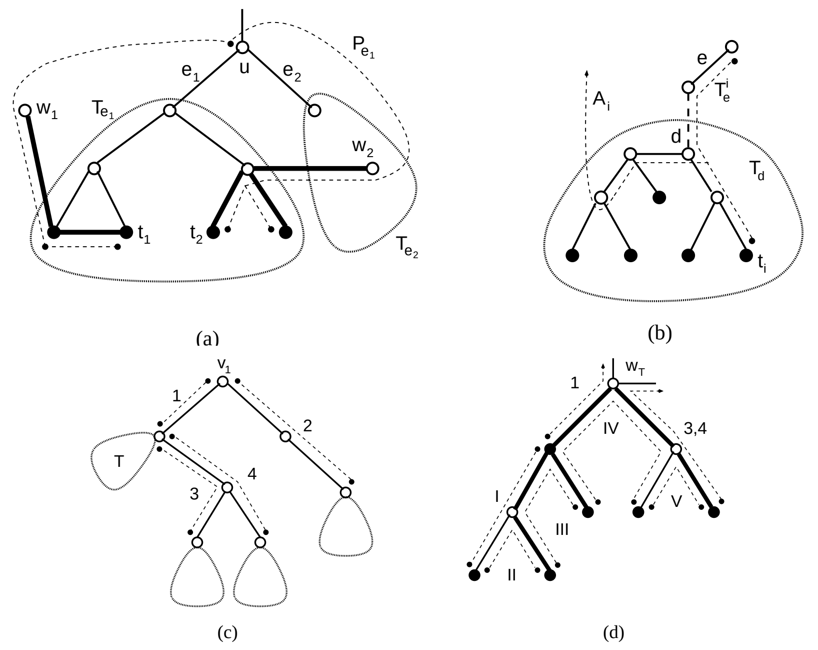

Fig. 1. (a), (b) Alternate trees and paths in PTCGs; (c) Distribution of hierarchical players for a parent player i, (d) distribution of personalized players in the component T needed only by i

completely by i and all edges of T^* outside T_u. Let T_{e_j} be a tree that is not completely part of A_h. Consider for each terminal t of h located in T_{e_j} the path from t to u in A_h. We denote this set of paths by P_{e_j}. Let P'_{e_j} be the set of subpaths from P_{e_j} containing for every $P \in P_{e_j}$ the first part between the terminal of h and the first node $w \notin T_{e_j}$. This node always exists because $u \notin T_{e_j}$, and it is in T^*, because T^* covers all nodes in G. The network $A_{e_j} = \bigcup_{P \in P'_{e_j}} P$ was considered as a feasible deviation when constructing the payments for $T_{e_j} + e_j$, as it connects every terminal in T_{e_j} to a node of $T^* \backslash T_{e_j}$. Furthermore, the payments of i were the same, hence the cost of A_{e_j} was equal. Using the assumption that u is the first node, for which T_u is not optimal, we know that $c(A_{e_j}) \geq c(T_{e_j} + e_j)$. So after substituting A_{e_j} by T_{e_j} and e_j in A_h, the new network is at least as cheap as $T_{e_j} + e_j$. To show that this new network is also feasible, suppose we iteratively remove a path $P \in P'_{e_j}$. Now there might other terminals, whose connections to u use parts of P. The last node w of P is the first node of P outside of T_{e_j}, and it stays connected to u as P is the first part of a path to u. All other nodes of P are in T_{e_j} and will be connected by T_{e_j} and e_j. Hence, all terminals affected by the removal of P will finally be reconnected to u. In this way A_h can be transformed into T^* without cost increase. This proves that T^* is optimal under c'_h, so Algorithm 1 finds Nash equilibria. □

Figure 1a depicts the argument. Paths from the set P_{e_1} are indicated by dashed lines. The subgraph A_{e_1} of A_h is drawn bold. A node w can be either completely outside T_u (like w_1 for t_1) or in another T_{e_j} (like w_2 for t_2). Replacing A_{e_1} by T_{e_1} yields a feasible network that is not more expensive.

Lemma 3. *The payments calculated by Algorithm 1 purchase T^*.*

The proof of this lemma can be found in Appendix A. Together with Lemma 2 this proves Lemma 1. □

Finally, we show how to use Lemma 1 to prove Theorem 1. Suppose we are given a PTCG with k players. At first we simplify the graph by constructing an equivalent *metric-closure game* with the same players, terminals and a complete graph G' on the nodes of T^*. Edge costs are equal to the cost of the shortest path in G. Then we use the algorithmic framework and argue by induction on the number of players. Assume that the theorem holds for any PTCG with $k-1$ players. Now consider step 3 for a game with k players. If there is no leaf player, we can feasibly pick any player i and let $p_i = 0$. Otherwise, if i is a leaf player, we assign her to pay as much as possible on T^i such that she has no incentive to deviate. If the reduced network after the framework iteration is optimal under the reduced cost for the remaining $k-1$ players, the theorem follows with the induction hypothesis. Here we use Lemma 1 and Algorithm 1 to make the argument. Introduce a global player h, who accumulates all players except i. Players h and i share a single source at a node s, and an equilibrium assignment for h represents an optimal solution for the remaining players after removal of i. Hence, our inductive step is proven by Lemma 1. In fact we do not even have to care about player h, but instead we use Algorithm 1 only to determine p_i. Naturally, when used in step 3 we must employ cost function c^{iter} for Algorithm 1. □

3 Approximation of Nash Equilibria

In this section we present an algorithm to calculate cheap approximate Nash equilibria in polynomial time. The algorithm sketched in proof of Theorem 1 is not efficient. Either we must provide the socially optimum network as input or we must construct the optimum deviation for the collective player h to improve the solution network. In any case this requires to solve an instance of the Steiner tree problem. Instead, in this section we use *connection sets* to construct polynomial time approximation algorithms.

Definition 3. *[4] A connection set S of player i is a subset of edges of T^i, such that for each connected component C in $T^* \setminus S$ either (1) there is a terminal of i in C, or (2) any player that has a terminal in C has all of its terminals in C.*

For any node without a terminal and degree 2 in T^* incident edges belong to the same connection sets. So for convenience we assume that T^* has no Steiner nodes of degree 2. If every player purchases at most α connection sets, the payments will form an α-approximate Nash equilibrium. Note that a subset of a connection set also is a connection set.

An algorithm for PTCGs In connection games with 2 terminals per player the edges of T^* can be partitioned into equivalence classes S_J, where e and f belong to the same class iff $J = \{j : e \in T^j\} = \{j : f \in T^j\}$. Each S_J forms a connection set for all players $j \in J$, which is maximal under the subset relation. We will say that connection set S_J is *needed* by J. In the following PTCGs we will only consider maximal connection sets and not explicitly mention a player. This information is given by the subtrees, in which the set is located. Furthermore, when tree connection

requirements are present, connection sets are contiguous. For a proof of this statement see Lemma 8 in Appendix B.
Our algorithm uses the framework. In step 3 it assigns a leaf player i to purchase 2 connection sets. If i is no leaf player, $p_i = 0$. If a leaf player i is removed in step 4, distinct connection sets might join and subsequently be wrongly regarded as a single connection set. This happens when they are needed by player sets differing only by i. We will use the notion of endangered sets to refer to these problematic sets.

Definition 4. *A connection set is called* endangered set *for player i if it is needed by the set of players $J \cup \{i\}$, and there is another connection set (called* forcing set*) needed by the set J, with $i \notin J$.*

Lemma 4. *For any leaf player in a PTCG there are at most two endangered sets.*

A proof of this Lemma is found in Appendix B. We will denote the endangered set with empty forcing set as the *personalized set*, the endangered set with nonempty forcing set as the *community set*. In step 3 we simply assign a leaf player to purchase these sets. It requires an easy inductive argument to show that the algorithm then works correctly. This yields the following theorem.

Theorem 2. *For any optimum centralized solution T^* in a PTCG, there exists a 2-approximate Nash equilibrium such that the purchased edges are exactly T^*.*

For PTCGs we know that T^* is connected, hence we can use a 1.55-approximation algorithm for the Steiner tree problem [18] to get an initial approximation T. Furthermore, we can use polynomial time shortest-path algorithms to find deviations and connection sets of optimum cost. Using a trick of Anshelevich et al [4] we can iteratively improve T by exchanging connection sets with better paths. For polynomial running time we must ensure substantial cost reduction for each exchange, which can be done by an adjustment of the edge cost. The details are deferred to the full version of this paper.
Thus, for PTCGs there is an algorithm that finds a $(2+\epsilon)$-approximate Nash equilibrium on a 1.55-approximate network in time polynomial in n and ϵ^{-1}, for any $\epsilon > 0$.

An algorithm for TCGs Next we adjust our algorithm to deliver 2-approximate Nash equilibria puchasing T^* for TCGs with any number of terminals per player. Each player (denoted as parent player) is divided into a set of child players with 2 terminals per player. Terminals of the child players are located at the same nodes as the ones of the parent player. In addition terminals of child players are distributed such that they create a PTCG. Hence, the set of all child players can purchase T^* with 2 connection sets per player.
The algorithm again uses the framework, and in step 3 a special procedure to assign the cost of T^* to the parent player i. First player i is divided into child players. Then child players of i are iteratively assigned to purchase endangered sets and removed. In the end i has to purchase all edges assigned to her child players. To identify personalized and community sets for the child players of i, one might first create a PTCG by splitting all other parent players. However, the next lemma ensures that the assignment of personalized and community sets for a leaf parent player i does not depend

on the splitting of the other parent players. Hence, in step 3 we assign the edges without explicitly splitting other players than i. Details can be found in Appendix C.

Lemma 5. *The endangered sets of child players of a leaf parent player i are independent of the division of other parent players.*

In the remainder of the section we show how to divide a parent player i such that the union of connection sets purchased by her child players forms 2 connection sets. We get a 2-approximate Nash equilibrium purchasing the optimum network T^* by using a special splitting in *hierarchical*, *personalized* and *superfluous* child players.
At first we disregard all non-lonely terminals but one. Let the node carrying the last remaining non-lonely terminal t be node v_1. If the player has only lonely terminals, we pick an arbitrary terminal as t. Then consider T^* rooted at v_1. Once we arrive at an edge e that is needed only by i, the tree connection requirements allow us to argue that the whole subtree below e is also needed only by i. Here we insert child players in a hierarchical fashion. We contract all edges that are needed only by i. Let this adjusted tree be denoted T' and consider it in BFS-order rooted at v_1. For each node v carrying a terminal of i, we introduce a new child player. She has a terminal at v and the nearest ancestor of v in the tree carrying a terminal of i (see Figure 1c). These child players will be termed *hierarchical players*.
Second, we consider the portions of the tree that were contracted to form T'. For each maximal connected subtree $T \subset T^i$ that is needed only by i, let w_T be the root node that T shares with T'. Let player j be the first hierarchical child player, whose terminal t_j was placed at w_T. This player connects upwards in the tree. Now we consider T in DFS-order and locate t_j at the first node carrying a terminal of i. For each new node w_z carrying a terminal encountered in the DFS order, we introduce a new child player and locate her terminals at the nodes w_{z-1} and w_z. At any time there is only one lonely terminal in T. Finally, when the last node w_l carrying a terminal of i is reached, we move all remaining terminals at w_T to w_l. They belong to the hierarchical players connecting downwards in the tree. Child players introduced in the DFS-scan of the components T are called *personalized players*, because they divide parts needed only by i (see Figure 1d).
Third, for every non-lonely terminal of i disregarded in the beginning, we introduce a *superfluous* child player connecting the terminal to v_1. They will not be assigned any payments.

Theorem 3. *For any optimum centralized solution T^* in a TCG, there exists a 2-approximate Nash equilibrium such that the purchased edges are exactly T^*.*

Proof. A certain elimination order of child players is used. In any iteration a leaf player is picked, first pick the superfluous players and then in a bottom-up fashion to v_1 the personalized and hierarchical players. One connection set for the parent player is formed by the union of all personalized sets for the child players. The other connection set is the union of the community sets. Actually, a slightly stronger statement holds.

Lemma 6. *If the child players of a parent player i are created and eliminated in the described way, the removal of the personalized and community sets will only create components carrying terminals of i, respectively.*

The proof of this lemma can be found in Appendix C. The removal of any one of the connection sets creates only components with terminals of i. Consider single connection sets assigned for players in later iterations of the algorithmic framework on a reduced network. With the property of Lemma 6 we know that single connection sets for the reduced network are also single connection sets for the original network. This completes the proof of Theorem 3. $\square$

In the full version of this paper we will show how to combine polynomial time approximation algorithms with our scheme to derive a $(3.1+\epsilon)$-approximate Nash equilibrium on a 1.55-approximate network. The assignment procedure used by ADTW is also analyzed by connection sets, however, it does not employ so much structural information as our child player splitting. The tree connection requirements allow us to use a splitting of parent players and proceed in a hierarchical elimination order. This avoids a matching step employed by ADTW to generate the assignment of edge costs to players. This is crucial for achieving a guarantee of 2 connection sets.

A tightness argument For every connection game ADTW finds 3-approximate Nash equilibria purchasing T^* given only the optimum network T^* as input. Like our algorithm it uses connection sets and does not employ cost sharing of edges. Unfortunately, any deterministic algorithm using only T^* as input cannot improve upon these algorithms even if it uses cost sharing. In this way ADTW for general connection games and our algorithm for TCGs represent optimal algorithms. Our algorithm, however, provides better worst-case performance on TCGs, because ADTW eventually assigns 3 connection sets to one player of a TCG with a cheap alternative strategy. Proofs of the following three theorems are presented in Appendix D.

Theorem 4. *For any $\epsilon > 0$ there is a connection game such that any deterministic algorithm using only the optimum solution T^* as input constructs a payment function, which is at least a $(3-\epsilon)$-approximate Nash equilibrium.*

Theorem 5. *For any $\epsilon > 0$ there is a TCG such that any deterministic algorithm using only the optimum solution T^* as input constructs a payment function, which is at least a $(2-\epsilon)$-approximate Nash equilibrium.*

Theorem 6. *For any $\epsilon > 0$ there is a TCG such that ADTW constructs a $(3-\epsilon)$-approximate Nash equilibrium.*

4 Backbone Games

In this section we present the *backbone game*, an extension of the connection game to groups of terminals. Each of the k players has a set of groups of terminals. Each terminal may be located at a different node. The player strives to connect at least one terminal from each of her groups into a connected network. Different terminals may be located at the same nodes. Some important results from [4] translate directly to the backbone game by restriction to the connection game. The price of anarchy is k, and the price of stability $k-2$. It is *NP*-complete to decide, whether a given game has a Nash equilibrium, and there is a lower bound of $\left(\frac{3}{2}-\epsilon\right)$ on approximate Nash equilibria purchasing T^*. Finding the optimum network for a single player is the network design problem of the Group Steiner Tree (GSTP) [11, 17]. The problem of finding a centralized optimum solution network T^* generalizes the GSTP in terms of forest connection requirements, so we term this the *Group Steiner Network Problem (GSNP)*. There are polylogarithmic approximation algorithms for the GSTP, but we are not aware of any such results for the GSNP. Hence, we will concentrate on algorithms for games, in which the solution is guaranteed to be connected. The general case is left as an interesting direction for future work.

Single Source Backbones In a SBG each player i has a group $\mathcal{G}_i$ of g_i terminals and must connect at least one terminal to a given source node s. Note that the price of anarchy is still k as the example establishing the bound is a single source game. The price of stability, however, is 1, and cheap approximate equilibria can be found in polynomial time. Proofs of the following theorems can be found in Appendix E.

Theorem 7. *The price of stability in the SBG is 1.*

Theorem 8. *For the SBG there is a polynomial time algorithm to find a $(1+\epsilon)$-approximate Nash equilibrium purchasing a network T with $c(T)/c(T^*) \in$ $O(\log n \log k \log(\max_i g_i))$.*

Surprisingly the deviation factor from the connection game translates to the backbone game – in contrast to approximation factors of the Steiner tree problem. The construction and the derived results extend to SBGs on directed graphs and games, in which each player has a threshold on her maximum contribution and rather stays unconnected if her assigned share exceeds this threshold. These results translate from ADTW-SS and connection games. In a backbone game with a single source group $\mathcal{S}$ consider edges between the terminals of $\mathcal{S}$. No player will include them into her best deviations. Furthermore, they will not appear in the optimum forest T^*. Hence, we can construct an equivalent single source game with a source node s by introducing and contracting edges between all nodes with terminals from $\mathcal{S}$.

5 Conclusion

In this paper we presented existence proofs and algorithmic results for finding exact and approximate pure strategy Nash equilibria in a network creation game. Our tree connection game is a variant of the connection game [4], in which every feasible solution network is guaranteed to be globally connected. For the special case of 2 terminals per player the price of stability is 1, i.e. there is a Nash equilibrium as cheap as the optimum network. Unfortunately, the algorithm constructing this equilibrium has exponential running time. Instead we derived a polynomial time algorithm to find $(3.1+\epsilon)$-approximate Nash equilibria on a 1.55-approximate network, even for tree connection games with any number of terminals per player. These quality guarantees improve significantly over ADTW, the only previous algorithm for this problem. Finally, we extended the connection game to the backbone game, in which players need to connect groups of terminals. For the single source case the price of stability is 1, and $(1+\epsilon)$-approximate Nash equilibria can be found in polynomial time.

Future work includes finding algorithms for approximate equilibria in general backbone games. Furthermore, there still is the potential for algorithms for approximate equilibria in TCGs and general games with improved approximation guarantees.

References

1. A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM J Comp*, 24(3):445–456, 1995.
2. S. Albers, S. Eilts, E. Even-Dar, Y. Mansour, and L. Roditty. On nash equilibria for a network creation game. In *Proc 17th Ann ACM-SIAM Symp Discrete Algorithms (SODA)*, 2006. to appear.
3. E. Anshelevich, A. Dasgupta, J. Kleinberg, É. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. In *Proc 45th Ann IEEE Symp Foundations Comp Sci (FOCS)*, pages 295–304, 2004.
4. E. Anshelevich, A. Dasgupta, É. Tardos, and T. Wexler. Near-optimal network design with selfish agents. In *Proc 35th Ann ACM Symp Theo Comp (STOC)*, pages 511–520, 2003.
5. V. Bala and S. Goyal. A non-cooperative model of network formation. *Econometrica*, 68:1181–1229, 2000.
6. C. D. Bateman, C. S. Helvig, G. Robins, and A. Zelikovsky. Provably good routing tree construction with multi-port terminals. In *Proc 1997 ACM/SIGDA Intl Symp on Physical Design (ISPD)*, pages 96–102, 1997.
7. J. Corbo and D. Parkes. The price of selfish behavior in bilateral network formation. In *Proc 24th Ann ACM Symp Principles of Distributed Comp (PODC)*, 2005.
8. A. Czumaj, P. Krysta, and B. Vöcking. Selfish traffic allocation for server farms. In *Proc 34th Ann ACM Symp Theory Comp (STOC)*, pages 287–296, 2002.
9. A. Fabrikant, A. Luthera, E. Maneva, C. Papadimitriou, and S. Shenker. On a network creation game. In *Proc 22nd Ann ACM Symp Principles of Distributed Comp (PODC)*, pages 347–351, 2003.
10. J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proc 35th ACM Symp Theory Comp (STOC)*, pages 448–455, 2003.
11. N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the Group Steiner tree problem. *Journal of Algorithms*, 37:66–84, 2000.
12. M. Goemams and D. Williamson. A general approximation technique for constrained forest problems. *SIAM J Comp*, 24(2):296–317, 1995.
13. H. Haller and S. Sarangi. Nash networks with heterogeneous agents. Technical Report Working Paper Series, 2003-06, Lousiana State University, 2003.
14. M. Hoefer and P. Krysta. Geometric network design with selfish agents. In *Proc 11th Conf on Comp and Comb (COCOON), LNCS 3595*, pages 167–178, 2005.
15. M. Jackson. A survey of models of network formation: Stability and efficiency. In G. Demange and M. Wooders, editors, *Group Formation in Economics; Networks, Clubs and Coalitions*, chapter 1. Cambridge University Press, Cambridge, 2004.
16. E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proc 16th Ann Symp Theoretical Aspects Comp Sci (STACS)*, pages 404–413, 1999.
17. G. Reich and P. Widmayer. Beyond Steiner's problem: A VLSI oriented generalization. In *Proc 15th Intl Workshop on Graph-Theoretic Concepts Comp Sci (WG)*, volume 411 of *LNCS*, pages 196–210, 1989.
18. G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *Proc 10th Ann ACM-SIAM Symp Discrete Algorithms (SODA)*, pages 770–779, 2000.
19. T. Roughgarden and É.Tardos. How bad is selfish routing? *J ACM*, 49(2):236–259, 2002.
20. A. Schulz and N. Stier Moses. Selfish routing in capacitated networks. *Math Oper Res*, 29(4):961–976, 2004.
21. A. Vetta. Nash equilibria in competitive societies with application to facility location, traffic routing and auctions. In *Proc 43rd Ann IEEE Symp Foundations Comp Sci (FOCS)*, page 416, 2002.

Appendix

A Price of Stability for PTCGs

A.1 Proof of Lemma 3

Suppose e is the first edge, which cannot be paid for. Then there exists an alternative path A_i for player i and an alternative tree A_h for player h whose cost bounds the contributions. Consider any terminal t_j of player h, and let T^j and A_j be the paths between s and t_j in T^* and A_h, respectively. The following lemma about the structure of A_h is a generalization of [4, Lemma3.4], which holds directly for the path A_i for player i.

Lemma 7. *There is a minimum cost alternative tree A_h for player h with the following property. For any t_j there are two nodes v_j and w_j on A_j such that all edges on A_j from t_j to v_j are in T^j, all edges between v_j and w_j are in $E \backslash T^j$, and all edges between w_j and s are in $T^* \backslash T_e$.*

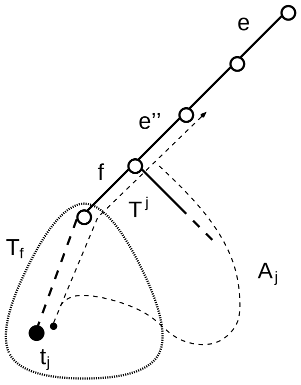

Fig. 2. A violation of Lemma 7

Proof. We will see that once A_h violates this lemma, it can be changed into a tree for player h that satisfies the properties of the lemma and is not more expensive. The proof follows closely the ideas of the proof of Lemma 2.
Once A_j reaches a node outside T_e, there is a connection of cost 0 to s, because all nodes from the graph are in T^*. Hence, in this case we can adjust A_h to satisfy the lemma without cost increase.
Now suppose A_j leaves T^j to $T_e \backslash T^j$ and re-enters T^j below e at another node. Consider the set of edges $e' \in T^j \cap A_j$ of T_e such that A_j excludes edges from $T_{e'}$. Let e'' be the deepest edge of this set in the BFS-traversal of T^* from s. Consider the edge $f \notin A_j$ directly below e'' on T^j (see Figure 2). Recall our assumption that e was the first edge that could not be paid for. By the time the algorithm was trying to purchase $T_f + f$, it found that the contribution of player h to T_f was optimal to connect all terminals of h in T_f to $T^* \backslash T_f$. As the payment functions and the adjusted cost function c'_h are built adaptively, we know that this is still true in the present iteration. We

can use the repairing construction from Lemma 2 to replace the respective parts of A_h with $T_f + f$. This yields a new feasible network that is not more expensive and uses all edges of T^j from t_j to e''. Hence, in the new network e'' is not considered anymore, and we move up to the next violating constellation. In this way A_h can be transformed without cost increase into a network obeying the lemma. □

Now we can argue that the payments assigned by Algorithm 1 pay for e. Let T_e^i be the part in T_e of the path between s and t_i. First suppose there is an edge $f \in A_h \cap T_e^i$. Then for a node v incident to f, A_h includes all edges from $T^h \cap T_v$, especially the node where T_e^i joins T^h. If player h deviates to A_h and player i sticks to her payments, this yields a feasible network with cost less than or equal $c(A_h) + c(A_i)$. Otherwise, assume that A_h totally excludes edges from T_e^i. A_i deviates from T_e^i at the *deviation point* d. However, as e is the first edge, which cannot be paid for, it is optimal for h use the contribution to T_d to connect all her terminals located in T_d to d. A_h can be transformed into a network including T_d without cost increase. By assumption d is not part of A_h, so the repaired network T_d might be (part of) a component, which is not connected to s anymore. This connection is then established by A_i. Figure 1b depicts this constellation. The structure of A_h ensures that the other terminals of h outside T_d will still be connected either to s or to T_d. Hence, if the contributions of i to A_i and h to A_h are not enough to pay for $T_e + e$, there is a cheaper network than T^* that can be constructed in one of the two ways described - a contradiction. This proves Lemma 3. □

B Approximate equilibria for PTCGs

B.1 Proof of Lemma 4

We will show that there is at most one endangered set with $J \neq \emptyset$. At first, we will see that in a PTCG all connection sets are contiguous. Note that we are considering only connection sets that are maximal w.r.t. the subset relation.

Lemma 8. *In a PTCG all edges of the same connection set form a contiguous path in T^*.*

Proof. Recall that in a PTCG every edge in such a connection set belongs to the same player subtrees. Suppose there is a connection set S, whose edges are not contiguous. Consider the path between two edges $e, f \in S$. There are edges from another connection set S' on the path between e and f. So the subtree T^j for a player j must be present to ensure S' is different from S. Suppose we remove e and f, then j remains connected. Then there are three components and one of them (denoted C_j) contains the terminals of player j. There is no player j' that establishes a connection requirement between C_j and the other components. As she had only 2 terminals, there can only be either e or f in $T^{j'}$ but not both. This, however, is not possible, because e and f are present in the same set of player subtrees by assumption. Hence, there is no tree connection requirement to connect C_j and the other components. This is a contradiction, because tree connection requirements were assumed. Note that in the assumed absence of non-terminal nodes of degree 2 each connection set is a single edge. □

We now assume for contradiction that for a leaf player i there are more connection sets with the stated property in Lemma 4 and pick arbitrarily two sets S_1' and S_2' with player sets J_1 and J_2, resp. The corresponding sets including i are denoted S_1 and S_2. We denote by $J = J_1 \cup J_2$ and assume $J_1 \neq \emptyset$ and $J_2 \neq \emptyset$.
Upon removal of S_1 and S_2 three components evolve. Two of them (denoted C_1 and C_2) each contain one terminal of i. As the subtree T^j for each player is a path, the third component (denoted C_3) contains a terminal of each player in $(J_1 \cup J_2) - (J_1 \cap J_2)$. Neither S_1' nor S_2' can be located in C_3, so they must be located in C_1 or C_2.

Case 1: S_1' and S_2' are located in different components. Hence, after removal of S_1' and S_2' components C_4 and C_5 evolve (see Figure 3a). Now all terminals of players in J are distributed on C_3, C_4 and C_5. Hence, when connecting these into a new component and C_1, C_2 into a second component we get a feasible forest. This is a contradiction to the presence of tree connection requirements.

Case 2: Now both S_1' and S_2' are located in the same component. Hence the other component holds a terminal of each of the players in one set, w.l.o.g. we assume C_1 a terminal from each player in J_1. Then, as S_1' is in C_2 all players of J_1 need both S_1 and S_2, so $J_1 \subset J_2$. Hence, in C_3 there is one terminal of each player of $J_2 - J_1$. In C_2 there is only one terminal of each player in J_1. When we remove S_2', we split off a new component C_4 containing one terminal of each of the players in J_2. S_1' is, of course, located in C_4, because it is needed by a subset of the players. If we remove S_1', we get a new component C_5 with a terminal of each of the players in J_1 (see Figure 3b). In C_4 one terminal of each of the players in $J_2 - J_1$ remains. So if we connect C_4 and C_3 into a component, there is no need to connect this new component to the rest of the tree and the presence of tree connection requirements is violated.
Hence, it is shown that there is only one endangered set with $J \neq \emptyset$. In addition for $J = \emptyset$ there is also just one connection set. This yields at most two endangered sets for a leaf player and the lemma is proven. $\square$

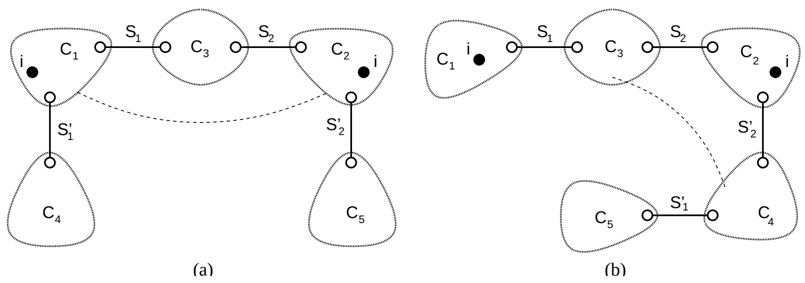

Fig. 3. Different cases with components violating tree connection requirements

C Approximate equilibria for TCGs

C.1 Proof of Lemma 6

To see the argument, we first have a closer look at the structure of endangered and forcing sets.

Lemma 9. *For any leaf player in a PTCG the personalized, community and the non-empty forcing sets share a common node if they exist.*

Proof. Suppose for a leaf player i the sets do not share a node. If there is no nonempty forcing set, there is only the personalized set S_i and the lemma follows trivially. So let there be a nonempty forcing set S_J needed by player set J. Remove the community set, and let C_1, C_2 be the components with and without the lonely terminal of i, resp.

Case 1: Let S_J be in C_2 and remove it. This splits C_2 into two components. We will denote by C_2' the remaining component including the terminal of i. The other component is denoted by C_3, and it carries one terminal of each player in J. Now remove all edges that connect to the node v carrying the lonely terminal of i in C_1. Connect all resulting components except for v to C_3. Then connect the node v to C_2'. All connection requirements will be met, but there is a solution with two components. This violates the tree connection requirements. Hence, we know that S_J must be in C_1.

Case 2: Let S_J be in C_1 and remove it. Similar to Case 1 we will speak of C_1' and C_3 as resulting components after removal of S_J. Now suppose there is another player j with a terminal located in C_1'. T^j can only include one of S_J and $S_{J\cup\{i\}}$, so $j \notin J$, and she must have both her terminals in C_1'. Now isolate the lonely terminal of i on node v again. Then construct two components, one consisting of v, C_2 and C_3; the other consisting of all other components. This will generate a feasible solution with two components, which is a violation of tree connection requirements. It is easy to observe that once there are no terminals in C_1', there can be no connection sets in C_1', except for the personalized set S_i needed only by i. The only thing left to show is that S_i must be located in C_1'.

Case 3: Suppose S_i is in C_2 and remove it. Again we denote C_2' the component with the terminal of i and C_3 the other one. Observe that C_3 must carry all terminals of J. Then remove S_J from C_1 generating C_1' and C_3'. We can isolate the components carrying terminals of J from the rest of the components. Hence, if C_3 and C_3' are combined and C_1' and C_2' are combined, a solution with 2 components is possible. This again violates the tree connection requirements.

Hence, if S_i is present in C_1', the three connection sets share a Steiner node. Otherwise, the two connection sets meet at the node carrying the lonely terminal of i. This concludes the proof of Lemma 9. □

Now, to make our inductive argument, we will reverse the sequence of player and edge removals and instead consider player, edge and node insertions. We only present

the inductive argument. Suppose we have already created a network with players, for which the assumption holds. Now we are processing parent player i and enter one of her child players j. This player can have a personalized set. For the union of all personalized sets of previous child players of i the assumption holds that after removal every component contains a terminal of i. If in addition the newly added personalized set is removed, the only additional component that evolves is the node carrying the lonely terminal of j. Hence, the lemma is proven for the personalized sets.
For the community sets, the case is more complicated. At first we add the class of superfluous child players, but they do not purchase anything and thus do not violate the argument. Next, consider the hierarchical players. Their community sets are always needed by at least one additional player, which is not a child player of i. Consider only these community sets, then after removal every component has a terminal of i. This is ensured by the hierarchical structuring and the point that other players are also present on the sets. Now we also remove the community sets introduced by the personalized players. Consider a subtree T described in the creation of personalized players, which is needed only by i. Note that the root vertex w_T of T does not have to carry any terminals of i. In addition it might be adjacent to a couple of community sets of hierarchical players. Note that the whole subtree T must be purchased by i, hence it consists completely of personalized and community sets of child players of i. To satisfy the lemma every node in T must be connected by a (possibly empty) path of personalized sets to a node with a terminal of i. In addition, the root w_T must be connected by personalized sets to the node carrying the terminals of the hierarchical players connecting downward in the tree. This serves to keep the previous argument for hierarchical players feasible. Due to the structure shown in Lemma 2 for the personalized and community sets a new community set is only introduced when a node without a terminal is created. In this case, however, the lonely terminal of the corresponding personalized player is to the lower right of the node, because the players were created with a DFS-order. Hence, the root of any subtree of T is connected by personalized sets to the rightmost terminal of the subtree, which is the last one from the subtree in the DFS-order. For illustration see Figure 1d, in which the bold lines indicate the personalized sets of the child players I-V. This proves the case for community sets.
The splitting for a leaf parent player i creates connection sets, that upon removal yield only components with terminals of i in it. Hence, if a connection set is removed, all resulting components must be reconnected to form a feasible network. This is the key property that keeps the connection sets valid, even when the network is perturbed by adding other players. This proves Lemma 6. □

C.2 Proof of Lemma 5

Again we use the reversal of the induction by considering player, edge and node insertions. Suppose we have a child player j of parent player i, who is entered into the network. Let there be an arbitrary splitting of the other parent players into child players obeying the tree connection requirements. The personalized set of j is independent of the splitting of the other parent players. So suppose she has a community S_c set needed by $J \cup \{j\}$ and a forcing set S_f needed by J. Consider a different player

i' with a child player in J and $J \cup \{j\}$. $T^{i'}$ cannot have a terminal at v and must not include any other edges incident at v than the two edges in S_c and S_f. Otherwise the tree connection requirements would require a different set of child players of i' on S_c and S_f, which would contradict the assumption that a community set for j is present. So let us now assume a different splitting of i'. We get different player sets needing S_c and S_f. There must be at least one child player of i' needing the sets, because they are both in T^i. However, as there is no terminal or alternate connection at v, any child player of i' needing S_c will also need S_f. Hence, S_c remains the community set for j. This argument can easily be adjusted for more players. $\square$

D Tightness Results

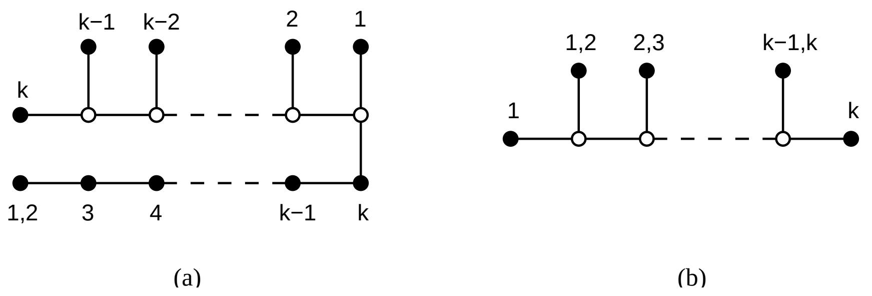

Fig. 4. Tightness example games for the algorithms

D.1 Proof of Theorem 4

Consider a game with 2 terminals per player and an optimal solution T^* of cost $3k-3$ shown in Figure 4a. All edges of T^* have cost 1. There is at least one player that pays a cost of $3 - \frac{3}{k}$. Each edge is a distinct maximal connection set, because it is shared by a distinct set of player subtrees T^i. One player can have an alternative path of cost $(1+\epsilon)$ outside T^*. As this path is not known to the algorithm, the best strategy is to equilibrate payments between players. It assigns each player i to pay a cost of $3 - \frac{3}{k}$ for parts inside her path T^i. As k approaches infinity, the worst-case deviation factor becomes at least $(3-\epsilon)$. $\square$

D.2 Proof of Theorem 5

For the game in Figure 4b there are exactly $2k-1$ connection sets of cost 1. It is optimal to let each player pay a cost of $2 - \frac{1}{k}$ inside her subtree. So as a deterministic algorithm working only with T^* our algorithm delivers the optimum asymptotical worst-case guarantee. $\square$

D.3 Proof of Theorem 6

Consider the game in Figure 4b. The algorithm proceeds as follows. At first each player purchases her personalized set. Then in the first step, it picks two terminals of a player and assigns the path to be purchased by terminals connecting to it. Afterwards the paths to the purchasing terminals are assigned and so on until T^* is purchased. The distribution of edges to terminals is done in a matching step. This is optimal for the general case, for TCGs however, it might yield an unlucky assignment. In our example it is possible, if the assignment starts by picking player 2, matching might assign the terminals of player 1 to purchase 2 connection sets. With the personalized set this results in 3 connection sets. If all edges have cost 1 and there is an alternative path of cost $(1 + \epsilon)$, player 1 has a deviation of factor at least $(3 - \epsilon)$. $\square$

E Single Source Backbones

E.1 Proof of Theorem 7

Consider T^* in BFS order from the source s. The algorithm constructing the payments considers edges in reverse order and with an adjusted cost function similar to the procedure for connection games. It assigns each player at most the cost of her cheapest deviation path – in this case between any of her group terminals and s. To show that this algorithm yields a Nash equilibrium purchasing T^*, we use a reduction to connection games. Construct a new graph $G' = (V', E')$ by adding an artificial node u_i for each player i. Let $U = \{u_1, \ldots, u_k\}$ and $V' = V \cup U$. Connect u_i to all nodes carrying terminals of player i with an artificial edge of cost $c(G)$. Let the set of artificial edges for player i be E_i and let $E' = E \cup (\bigcup_i E_i)$. The single source game in G', in which each player strives to connect u_i to s, is called the *corresponding connection game (CCG)*. The centralized optimum network T^* for the backbone game corresponds to a centralized optimum network T^*_c for the CCG and vice versa, as the degree of every u_i is 1 in T^*_c. Note that a cheapest deviation for a player in both games consists of a path. Applying ADTW-SS in the CCG every player i will get assigned exactly one artificial edge, and every reasonable deviation for player i in G' will also include only one such edge incident to u_i. Hence, there is a correspondence of deviation paths and the calculated Nash equilibrium for the CCG is also a Nash equilibrium for the backbone game. Furthermore, there is also a network improvement step possible, as in each improved network of the CCG all nodes u_i have degree 1. Hence, such a network yields a better feasible network in the backbone game. Finally, observe that our algorithm is equivalent to ADTW-SS in the CCG. $\square$

E.2 Proof of Theorem 8

Suppose we are given an α-approximate network T. We employ the adjustment presented for ADTW-SS and first reduce the cost of every edge by a cost of $\gamma = \frac{\epsilon c(T^*)}{(1+\epsilon)\alpha n}$ with $\epsilon > 0$. For the procedure constructing a Nash equilibrium on T^* there is also a repairing step similar to ADTW-SS. Hence, we can improve the solution until we find a network that can be assigned without incentives to deviate. With at most $\frac{(1+\epsilon)\alpha n}{\epsilon}$

repairing steps, the algorithm runs in time polynomial in n, α and ϵ^{-1}. Following the analysis in [4] we can argue that the algorithm determines a $(1+\epsilon)$-approximate Nash equilibrium. However, to ensure the constant $1+\epsilon$ and polynomial running time, we need for each game a *non-asymptotical* upper bound on α that is polynomial in n and k. We cannot hope for a constant, as GSTP generalizes set cover. Using an algorithm [6] to solve the GSTP with $\alpha = (1 + \ln \frac{k}{2})\sqrt{k}$ we can compare its solution against more recent efficient methods for the GSTP with asymptotical polylogarithmic performance guarantees [10, 11]. In this way the solution network will be an $O(\log n \log k \log(\max_i g_i))$-approximation, but never more than a $((1 + \ln \frac{k}{2})\sqrt{k})$-approximation of T^*. $\square$