Non-cooperative Tree Creation*

Martin Hoefer†

July 19, 2007

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]. Our study concerns an important subclass of *tree games*, in which every feasible network is guaranteed to be connected. It generalizes the class of single-source games considered in [4]. We focus on the existence, quality, and computability of pure-strategy exact and approximate Nash equilibria. For tree connection games, in which every player holds two terminals, we show that there is a Nash equilibrium as cheap as the optimum network. In contrast, for single-source games, in which every player has at most three terminals, the price of stability is at least $k-2$, and it is NP-complete to decide the existence of a Nash equilibrium. Hence, we propose polynomial time algorithms for computing approximate Nash equilibria, which provide relaxed stability and cost efficiency guarantees. For the case of two terminals per player, there is an algorithm to find a $(2+\epsilon, 1.55)$-approximate Nash equilibrium. It can be generalized to an algorithm to find a $(3.1+\epsilon, 1.55)$-approximate Nash equilibrium for general tree connection games. This improves the guarantee of the only previous algorithm for the problem [4], which returns a $(4.65+\epsilon, 1.55)$-approximate Nash equilibrium. Tightness results for the analysis of all algorithms are derived. Our algorithms use a novel iteration technique for trees that might be of independent interest.

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 in game theory and computer science 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. In many situations, however, a central institution interested in optimizing social desiderata has some means of controlling agent behavior. Hence, for such an institution it is of interest to know the boundary between efficiency and stability of the outcomes of the game. This trade-off is characterized by the price of stability [3], which is the ratio of the cost of the best Nash equilibrium over the cost of a social optimum solution. It captures *how efficient stability can get*, and has been studied in routing and network creation games [3, 4, 15, 20]. The more prominent measure is the price of anarchy [17] describing the cost of the worst instead of the best Nash equilibrium. It has received attention in networking problems, for instance routing [19], facility location [21] and load balancing [8, 17]. In this paper we consider these measures for 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 in a

*This work has appeared in part as an extended abstract at the 31st Symposium on Mathematical Foundations of Computer Science (MFCS 2006) [14] and the 17th International Symposium on Algorithms and Computation (ISAAC 2006) [13]

†Department of Computer & Information Science, Konstanz University, Box D 67, 78457 Konstanz, Germany. hoefer@inf.uni-konstanz.de. Phone: +49 7531 88 4263, Fax: +49 7531 88 3577. Supported by DFG Research Training Group 1042 "Explorative Analysis and Visualization of Large Information Spaces".

graph $G = (V, E)$, i.e. she holds a number of terminals at some nodes and wants to connect these nodes into a component. Each edge e has a non-negative edge cost $c(e)$, and each agent p picks as a strategy a payment function s_p that specifies for each edge e her money offer to the purchase of e. If the sum of all agent 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 or not. Each agent tries to fulfill her connection requirement at the least possible cost. A *state* in this game is a collection of strategies, one for each player, and a *Nash equilibrium* is a state such that no player has an incentive to unilaterally deviate from her strategy. An (α, β)-approximate Nash equilibrium is a state, such that each player can reduce her total payments by switching to another strategy by at most a factor of α, and which represents a β-approximation to the social optimum network. We refer to α as the *stability ratio* and to β as the *approximation ratio*.
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. 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.

1.1 Related Work

The connection game was introduced and studied in [4], where a variety of results were presented. The price of anarchy was shown to be precisely k and the price of stability at least $k - 2$. It is NP-hard to determine, whether a given game has a pure-strategy Nash equilibrium or not. There is a polynomial time algorithm that finds $(4.65 + \epsilon, 2)$-approximate Nash equilibria. For the single-source case, in which each player needs to connect a single terminal to a common source, a polynomial time algorithm finds $(1 + \epsilon, 1.55)$-approximate Nash equilibria. We denote these algorithms by ADTW-SS for the single source and ADTW for the general case. In [3] adjusted connection games were used 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 in [3, 7], e.g. on delays, weighted games and best-response dynamics.
Recently, connection games have been studied in a geometric setting in [15], which provided bounds on the price of anarchy and the minimum incentives to deviate from an assignment purchasing the social optimum network. The case of two players and two terminals per player was characterized in terms of prices of anarchy and stability, approximate equilibria and best-response dynamics.
More recently the mechanism of cost sharing considered for connection games in this paper was extended to resource sharing games in the context of facility location and integer covering problems [5, 13]. Some similar lower bounds in these games are NP-hardness of determining pure Nash equilibrium existence and prices of anarchy and stability of $\Theta(k)$. In these games, however, primal-dual approximation algorithms can be adjusted to find cheap approximate Nash equilibria in polynomial time.
A network creation game of different type was considered in [2, 6, 9, 11]. 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. See [16] for a recent overview of developments in the area of social network design games. 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.

1.2 Our Results

In this paper we will consider tree connection games (TCG). The games exhibit connection requirements such that every feasible solution network is connected. We analyze them with respect to exact and approximate pure-strategy Nash equilibria. We are especially interested in deterministic

polynomial time algorithms for deriving (α, β)-approximate Nash equilibria. This poses 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. We will not consider mixed-strategy equilibria in this paper, as they seem an unsuitable concept for a setting of network creation that requires concrete investments rather than randomized actions.

In Section 2 we show that for any tree connection game with two terminals per player the price of stability is 1. We outline an algorithm that allocates edge costs of a social optimum network to players such that no player has an incentive to deviate. This algorithm has exponential running time in general. However, it has polynomial running time if the social optimum network can be found efficiently. In particular, this is the case for any game with a constant number of players [10]. We show that our result is essentially tight by arguing that for games with at most three terminals per player it is NP-complete to decide if a game has a Nash equilibrium. The price of stability in these games increases to at least $k - 2$. These lower bounds are derived for the special case of single-source games, in which there is a source vertex to which every player wants to be connected.

In Section 3 we show how to distribute the cost of a social optimum network to get a $(2, 1)$-approximate Nash equilibrium for TCGs with any number of terminals per player. It can be translated into a polynomial time algorithm for $(2+\epsilon, 1.55)$-approximate Nash equilibria for TCGs with two terminals per player, and $(3.1 + \epsilon, 1.55)$-approximate Nash equilibria for TCGs with any number of terminals per player. The algorithm uses a recent 1.55-approximation algorithm for the Steiner Tree problem [18] as a subroutine, and the ratios improve if algorithms for Steiner Tree with better performance guarantees are found. Our algorithm improves over ADTW that provides $(4.65 + \epsilon, 1.55)$-approximate Nash equilibria.

In addition, we derive a tightness argument for the cost distribution technique used by our algorithm and ADTW. Both algorithms consider only the optimum network and use *connection sets* as building blocks for deriving approximate Nash equilibria. This construction allows a simple translation into polynomial time algorithms. We show that both our algorithm and ADTW are optimal with respect to the class of deterministic algorithms working only on the optimum network. This implies that methods with better performance guarantees must use ideas beyond connection sets and explicitly employ cost and structure of possible deviations. This significantly complicates their design and analysis. Nevertheless, finding improved algorithms seems possible and represents a challenging direction for future work. Finally, Section 4 concludes and presents furhter open problems and ideas for future research.

1.3 The Model

The *connection game* for k players is formally 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 p has a set $V_p \subset V$ of vertices that she strives to connect with a network. The sets V_p and V_q can overlap for different players p and q. We will call a vertex $v \in V$ a terminal or terminal vertex iff $v \in V_p$ for some player p, and a non-terminal or Steiner vertex otherwise. For terminals we frequently use the notation $t \in V$. A *strategy* for a player p is a function s_p, which specifies for each edge e the amount $s_p(e)$ that p offers for the purchase of e. If $\sum_{p=1}^{k} s_p(e) \geq c(e)$ for an edge e, it is considered *bought*. Bought edges can be used by all players to connect their terminals, no matter whether they contribute to the edge costs or not. A *strategy profile* or *state* s is a vector of strategies, which specifies for each player one strategy. The *individual cost* of a player is $u_p(s) = |s_p| = \sum_{e \in E} s_p(e)$ if s yields a network of bought edges between the terminals of p. Otherwise, we assume $u_p(s) = +\infty$, a prohibitively large cost. Thus, each player insists on connecting her terminals, because she rather purchases the social optimum network completely than accepts that her vertices are unconnected. A *Nash equilibrium* is a state such that no player can reduce $u_p(s)$ by unilaterally choosing a different strategy. A (α, β)-*approximate Nash equilibrium* is a state, in which each player can reduce $u_p(s)$ by at most a factor of α by switching to another strategy, and for which the purchased network represents a β-approximation to the social optimum network. We will use NE and (α, β)-NE as shorthands to refer to these concepts. The problem of finding a social optimum network for all players and an optimum strategy for a

single player are the classic network design problems of Steiner Forest [1, 12] and Steiner Tree [18], respectively. For the rest of this paper we will denote a social optimum network by T^*. The subtree of T^* that player p uses to connect her terminals is denoted by T_p.
We will deal with an interesting class of connection games, the *tree connection games (TCG)*, which are games with tree connection requirements.

Definition 1 *In a connection game there are tree connection requirements if for any two terminals* t_1 *and* t_{l+1}*, there is a sequence of players* $p_1, \dots, p_l$ *and terminals* $t_2, \dots t_l$ *such that player* p_j *the terminals* $t_j, t_{j+1} \in V_{p_j}$ *for* $j = 1, \dots, l$.

Tree connection requirements ensure that every possible solution network which satisfies all connection requirements is guaranteed to be connected.[1] A TCG can be thought of as a splitting of a single global player into k players, which preserves the overall connection requirements. For the subclass of TCG with two terminals per player we will use the term path tree connection game (PTCG). A different subclass are single-source games (SSG), in which there is one source terminal $s \in V_p$ for every player p

1.4 Initial Observations

Recall from [4] that a NE of the connection game has the following properties.

- The network of bought edges is a tree.
- For each edge e the total amount offered for its purchase is either $c(e)$ or 0.
- Each player contributes only to the subtree of bought edges that she needs to satisfy her connection requirement.

Our algorithms rely on a player-based assignment technique which might be of independent interest. It is presented in the following framework. In each iteration it picks a player, assigns payments, removes the player, and reduces the edge costs by the amount she paid. The framework terminates if there is no player left. As candidates for this elimination process it considers leaf players.

Definition 2 *A player owns a* lonely terminal $t \in V_p$ *if* $t \notin V_q$ *for any other player* $q \neq p$. *A player* p *in a TCG is a* leaf player *if she owns at least one lonely terminal and at most one non-lonely terminal.*

Algorithm 1: Algorithmic Framework

Input: A feasible tree T
Output: State $s = (s_1, \dots, s_k)$ distributing the cost of T

```
c^1(e) = c(e) for all e ∈ E
for iter ← 1 to k do
    p is a leaf player if possible; otherwise an arbitrary player
    Call a procedure, which either determines s_p or improves T
    if procedure returns a new tree then
        restart the framework with new tree
    Set c^{iter+1}(e) ← c^{iter}(e) − s_p(e) for all e ∈ E
    Remove p, contract edges e of cost c^{iter+1}(e) = 0
```

To provide some intuition how leaf players correspond to leaves, we consider the case of PTCG with two terminals per player. Consider a *connection requirement graph* G_{crg} constructed as follows.

[1] Clearly, there are games without tree connection requirements in which every feasible network is connected. In these games the structure of G forces players to build a connected network. For such games the results presented here do not necessarily hold.

The node set of G_{crg} contains all terminals from G. The edge set is created by introducing a single edge for each player between her terminals. Note that G_{crg} must be connected due to the presence of tree connection requirements. Suppose we run Algorithm 1 and apply the removal of players by removing the corresponding edge in G_{crg}. Whenever a non-leaf player is encountered and removed by Algorithm 1, we break a cycle in G_{crg}. Consider this set of edges corresponding to non-leaf players removed by Algorithm 1. This set of edges breaks all cycles in G_{crg}. Instead, we can remove this set of edges and corresponding players upfront and afterwards run Algorithm 1 on the resulting tree G_{crg}. This yields the same output of the algorithm. Then, in every iteration the algorithm is guaranteed to find a leaf player, which corresponds to an edge connecting a leaf node in G_{crg}. Note that in this case picking leaf players results in an order similar to an inverse BFS in G_{crg}.

2 Exact Nash Equilibria

In this section we examine cost and complexity properties of exact NE in TCGs. We first observe that the price of anarchy is as large as k, even in the PTCG. Consider a graph with two vertices and two parallel edges e_1 and e_2, in which each player wants to connect both vertices. Edge e_1 has cost k, e_2 cost 1. If each player is assigned to purchase a share of 1 of e_1, the state is a NE and the price of anarchy becomes k. Note that k is also an upper bound as was noted in [4]. In contrast, the price of stability for the PTCG is 1 as every such game has a NE purchasing T^*.

Theorem 1 *For any social optimum tree T^* in a PTCG there exists a Nash equilibrium exactly purchasing T^*. The price of stability in the PTCG is 1.*

Proof. We use Algorithm 1 with the optimum tree T^* as input to construct an optimal NE. In line 4 we use the following Procedure 2. As the optimum T^* is provided as input, the procedure only outputs a strategy s_p for each player p. The resulting algorithm is similar to ADTW-SS [4] for SSGs.

Procedure `ExactNash`(T^*, c, p) for computing a NE cost distribution of T^* for PTCGs

Input: The social optimum tree T^*, a cost function c and a selected player p
Output: A strategy s_p for p

1 Create global player h accumulating all players except p
2 Set $s_p(e) = s_h(e) = 0$ for all $e \in E$
3 **if** p has no lonely terminal **then**
4 **return** s_p
5 **if** p has no non-lonely terminal **then**
6 Set $s_p(e) = c(e)$ for all $e \in T_p$ and **return** s_p
7 Let s be the vertex with the non-lonely terminal of p
8 **for** each edge $e \in T^*$ in reverse BFS order from s **do**
9 **if** e is a bridge **then**
10 Assign $s_q(e) = c(e)$ to some player q, for which $e \in T_q$
11 **else**
12 **if** $e \in T_p$ **then**
13 Find the cheapest path A_p excluding e for player p under c_p^e
14 Assign $s_p(e) = \min(c(e),\ c_p^e(A_p) - s_p(T^e))$
15 **if** $e \in T_h$ **then**
16 Find the cheapest tree A_h excluding e for player h under c_h^e
17 Assign $s_h(e) = \min(c(e) - s_p(e),\ c_h^e(A_h) - s_h(T^e))$
18 **return** s_p

In the description T^e denotes the part of T^* rooted at s, which is located below edge e. When assigning the cost of e to player p the procedure uses cost functions c_p^e for G with $c_p^e(e') = s_p(e')$ for $e' \in T^e$, $c_p^e(e') = 0$ for $e' \in T^* \backslash T^e$ and $c_p^e(e') = c(e')$ otherwise. c_h^e is defined similarly. Note that the creation of player h (line 1) and building her function s_h (lines 15-17) have no influence on the output and can be dropped from the procedure. The consideration of player h, however, is useful for proving correctness of the algorithm.
We must show that Algorithm 1 using Procedure 2 yields a NE purchasing T^*. As a first observation, it is possible to create an equivalent game as follows. For the task of distributing the cost of T^* we can drop all vertices outside of T^* from consideration. Instead, we can consider the complete graph G_{T^*} on the vertices of T^*. Edge costs c_{sp} are determined by the cost of shortest paths in G with respect to c. Note that each player considers only paths as deviations. Furthermore, as no player is assigned a contribution to an edge outside T^*, a player has to purchase the full cost of every such edge she uses in a deviation. Hence, replacing G and c by G_{T^*} and c_{sp} does not alter the cost of best deviations at any point in the algorithm. Thus, in the following we assume that c satisfies the triangle inequality and that T^* is a minimum spanning tree of G.
Trivially, the algorithm works correctly for every PTCG with only one player. Hence, we assume as our induction hypothesis that the algorithm works correctly for every PTCG with $k-1$ players. Then consider a PTCG with k players. Our induction step is represented by the first iteration of the framework. We must show that the result of this iteration is a PTCG with $k-1$ players such that T^* is an optimum solution under the resulting cost function c^2. The induction hypothesis can then be used to argue that it allows a NE cost distribution for the remaining $k-1$ players. If player p has no incentive to deviate from s_p, a NE purchasing T^* evolves. Thus, it remains to show the following two properties for the induction step:

1. Strategy s_p assigned to player p allows her no cheaper deviation.
2. T^* is optimal for the remaining $k-1$ players under cost function c^2.

We attack the proof by further adjusting the game. All costs of edges in T^* not purchased by p must be purchased by some of the other players. Thus, we suppose that all players except p are represented by the global player h, who accumulates all terminals except those of p. Procedure 2 assigns costs to both players p and h. Naturally, if at the end of the procedure player h has no incentive to deviate from s_h, then property 2 is fulfilled. Hence, both properties are fulfilled if there is a NE purchasing T^* in the game for players h and p. It remains to prove the following lemma.

Lemma 1 *For the reduced game with two players p and h Procedure 2 computes a Nash equilibrium purchasing T^*.*

Proof. We show that at the end of the procedure no player has a possibility to lower her contributions by changing her strategy, and that the calculated strategies yield connections of bought edges - i.e. T^* is fully paid for. Note that the game actually represents a single source game for players p and h with a single source terminal s. For the rest of the proof we consider T^* rooted at s and use terms *higher* and *lower* to refer to edges and vertices at a closer and further distance from s in T^*, respectively.
We first focus on the question if a player can lower her contributions.

Lemma 2 *The payments computed by Procedure 2 allow no cheaper feasible deviation for players p and h.*

Proof. With the similarity of Procedure 2 to ADTW-SS the lemma follows directly from [4, Theorem 3.2] for player p. For player h, however, the argument is more complicated.
Recall that T^e denotes the part of T^* below edge e. We denote by T^u the part of T^* below a vertex u. Note that we assume $e \notin T^e$, but $u \in T^u$. Consider s_p and s_h at the end of an iteration, in which Procedure 2 has assigned the cost of some edge $e = (u, v)$ for which we assume u is higher than v. It is not obvious that the construction of c_h^e leads to an equilibrium strategy for player h. Consider a vertex u where multiple subtrees join. We assume that for each edge e_j

below u the contribution of h to $T^{e_j} + e_j$ is small enough, i.e. $T^{e_j} + e_j$ is the cheapest way under c^e_h to connect the terminals of h in T^{e_j} to $T^* \backslash T^{e_j}$. But player h owns terminals in possibly *all* subtrees T^{e_j}, and when constructing the payments for edges of a single $T^{e_j} + e_j$ the contribution to the respective other subtrees was considered to be 0. Can h pick a different, cheaper tree to connect her terminals in T^u to $T^* \backslash T^u$ that improves upon her calculated contribution? We give a negative answer to this question as follows. Assume the procedure assigned payments for each $T^{e_j} + e_j$, and that u is the first vertex at which $s_h(T^u)$ is not optimal for h. Cost function c^u_h with $c^u_h(e') = s_h(e')$ for $e' \in T^*$ and $c^u_h(e') = c(e')$ for $e' \in E \backslash T^*$ captures the optimization problem faced by player h. In fact, we will see that T^* is indeed the optimum network under c^u_h, hence player h sticks to her contribution.

Suppose u is the first vertex, at which player h can lower her contribution. Then there is a deviation network A_h, which is cheaper than T^* under c^u_h (i.e. the current contribution of h to T^*). W.l.o.g. A_h includes all edges of cost 0, in particular all edges purchased completely by p 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 $P_1^{e_j}$ 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 vertex $w \notin T^{e_j}$. This vertex always exists because $u \notin T^{e_j}$. It is in T^* because in our adjusted game T^* covers all vertices in G. The network $A_h^{e_j} = \bigcup_{P \in P_1^{e_j}} P$ 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 vertex of $T^* \backslash T^{e_j}$. Furthermore, the payments of p were the same, hence the cost of $A_h^{e_j}$ was the same. Using the assumption that u is the first vertex for which T^u is not optimal for h, we know that $c^u_h(A_h^{e_j}) \geq c^u_h(T^{e_j} + e_j)$. After substituting $A_h^{e_j}$ by $T^{e_j} + e_j$ in A_h, the new network is not more costly than $T^{e_j} + e_j$. To show that this new network is also feasible, suppose we iteratively remove a path $P \in P_1^{e_j}$. Then there might other terminals whose connections to u use parts of P. The last vertex w of P is the first vertex 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 vertices of P are in T^{e_j} and are connected by T^{e_j} and e_j. Hence, all terminals affected by the removal of P are finally reconnected to u. In this way A_h can be transformed into T^* without cost increase. This contradicts our assumption that A^h is cheaper than T^* and proves that T^* is optimal under c^u_h. Thus, player h cannot lower her contribution, which proves the lemma. □

Figure 1 depicts the argument for player h. Vertex u has two subtrees for which the payments were assigned independently. The subtree $A_h^{e_1}$ of A_h, which is assumed to be cheaper than T^{e_1}, is drawn in bold. A vertex w represents the first vertex outside T^{e_1} on a path in A_h, and it 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 $A_h^{e_1}$ by T^{e_1} and e_1 yields a feasible network that is not more expensive.

We have shown that the players have no incentive to move away from their assigned payments due to cost improvement. In addition, we need to show that the payments suffice to pay for the cost of T^*.

Lemma 3 *The payments computed by Procedure 2 purchase* T^*.

First, consider the structure of A_h in an iteration when assigning costs of edge e. Consider a 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 generalizes [4, Lemma 3.4] to player h with more than two terminals. It similarly holds for player p and her minimum cost deviation A_p.

Lemma 4 *Suppose edge* e *is the first edge that cannot be paid for using the assignment procedure of the algorithm. Then there is a minimum cost alternative tree* A_h *for player* h *with the following property. For any* t_j *there are two vertices* v_j *and* w_j *on* A_j *such that all edges on* A_j *from* t_j *to* v_j *are in* $T_j \cap T^e$*, all edges between* v_j *and* w_j *are in* $E \backslash (T_j \cap T^e)$*, and all edges between* w_j *and* s *are in* $T^* \backslash T^e$.

Proof. The proof is by contradiction. If A_h violates this lemma, it can be transformed into a tree for player h that satisfies the properties of the lemma and is not more expensive. Suppose edge e is the first edge that cannot be paid for. Consider the path A_j from a terminal t_j to s.

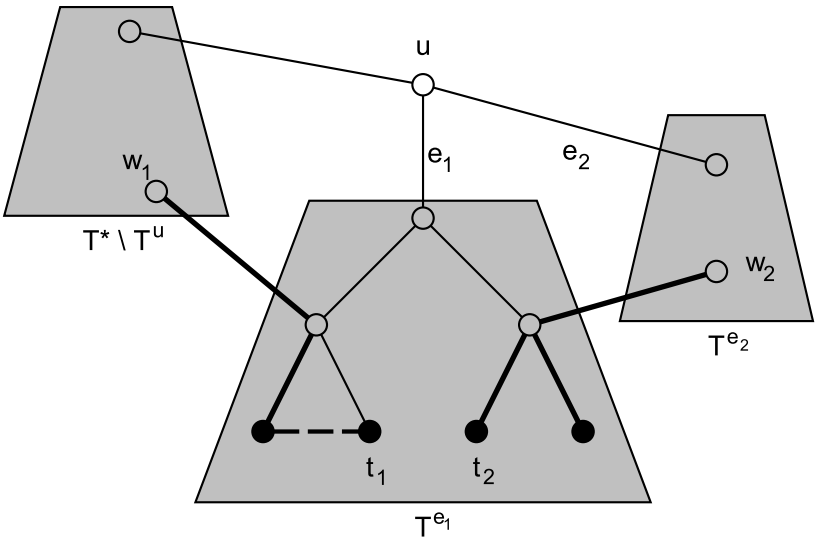


Figure 1: Transformation of $A_h^{e_1}$ into T^{e_1}. Solid edges in the marked area belong to T^{e_1}, bold edges to $A_h^{e_1}$. Replacing $A_h^{e_1}$ in A_h by T^{e_1} and e_1 yields a feasible network for player h that is not more expensive than A_h under c_h^u.

Once it reaches a vertex outside T^e, there is a connection of cost 0 to s, because all vertices from the graph are in T^*. 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 vertex. Consider the edges $e' \in T_j \cap A_j$ of T^e such that A_j excludes edges from $T^{e'}$. Let e'_{low} be the lowest of these edges. We denote by $f \notin A_j$ the edge directly below e'_{low} on T_j (see Figure 2(a)). 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^e 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'_{low}. Hence, in the new network e'_{low} is not considered anymore as one of the edges $e' \in T_j \cap A_j$, for which A_j excludes edges from $T^{e'}$. Thus, iteratively A_h can be transformed without cost increase into a network obeying the lemma. □

Note that the argument can be extended to see that the vertices v_j build a frontier in T_e in the sense that A_h does not use any edge of T^e above any of the vertices v_j. The vertices v_j are similar to the *deviation points* used in the proof for SSGs in [4].

Proof. (of Lemma 3) Now we prove that the payments assigned by Procedure 2 pay for e. Denote by T_p^e the part of the path between s and the lonely terminal t_p, which is located in T^e. We consider two cases and show in each case how to build a better network than T^* if Lemma 3 is violated.

Case 1: Suppose there is an edge $f \in A_h \cap T_p^e$. Then for a vertex v incident to f, A_h includes all edges from $T^h \cap T_v$, in particular the vertex where T_p^e joins T_h. If player h deviates to A_h and player p sticks to her payments, this yields a feasible network with cost less than or equal $c(A_h) + c(A_p) < c(T^*)$.

Case 2: Assume that A_h excludes all edges from T_p^e. Then A_p departs from T_p^e at some vertex d. However, as e is the first edge, which cannot be paid for, it is optimal for player h use the contribution to T^d to connect all her terminals located in T^d to d. A_h can be transformed

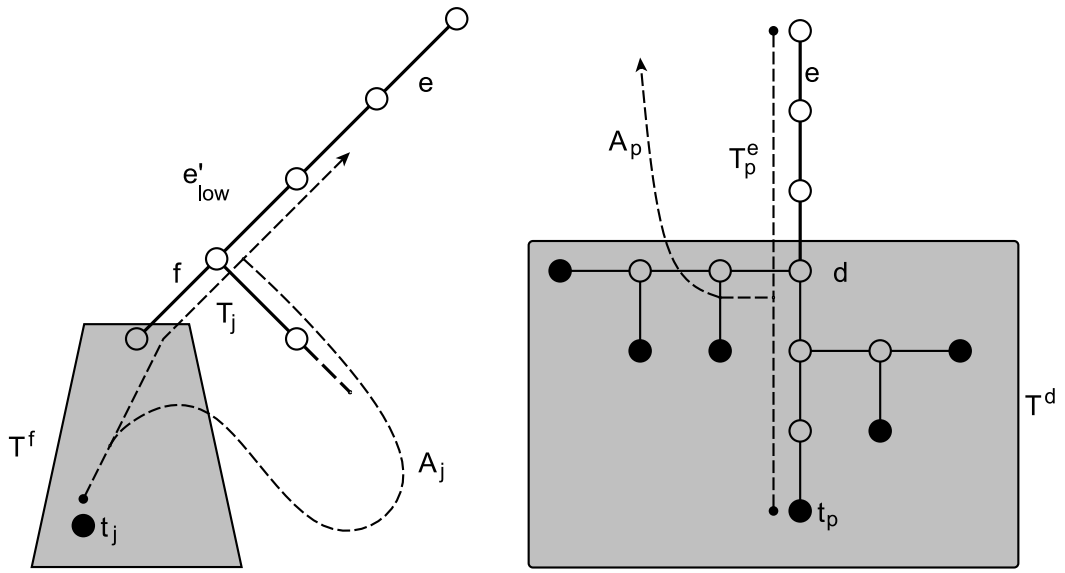


Figure 2: (a) A violation of the structure of Lemma 4, as A^j leaves and returns to T^e below e. As $T^f + f$ is the cheapest way to connect the terminals of h in T^f to $T^* \backslash T^f$, it is possible to replace A_j by T_j for every terminal $t_j \in T^f$ of h. This generates a feasible network that is not more expensive than A_h under c^h_e. (b) Improvement step for T^* if it cannot be paid for. T^d is optimal for h to connect all her terminals of T^d to d. Player p also sticks to her contributions in T^d and establishes a connection to s with A_p, which ensures feasibility.

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 the source s anymore. This connection is then established by A_p. Figure 2(b) depicts this constellation. The structure of A_h ensures that the other terminals of h outside T^d are still connected either to s or to T^d. Note that if A_p uses other edges of T^e outside A_h, we must ensure that h makes her contribution to these edges, because otherwise the cost for p is increased. However, we can again apply the same arguments to transform A_h such that a feasible network without cost increase is created.

Hence, if the costs $c^e_p(A_p)$ and $c^e_h(A_h)$ do not allow to pay for $T^e + e$, there is a cheaper network than T^* that can be constructed in one of the two ways described. This is a contradiction and proves Lemma 3. □

The previous lemmas show that no player has a possibility to lower her contributions and that the cost of T^* is paid for. This proves Lemma 1. □

As Lemma 1 captures the crucial part of our induction, Theorem 1 follows. This proves that the price of stability in the PTCG is 1. □

If the Algorithm 1 uses Procedure 2 with T^*, then it computes an optimum NE. For classes of PTCGs posing efficiently solvable subclasses of Steiner Tree (e.g. for constant k [10]), a NE purchasing T^* can be computed in polynomial time. Unfortunately, these advantageous properties are restricted to the PTCG. In a SSG with at most three terminals per player it is NP-hard to decide, whether the game has a NE or not. Furthermore, the price of stability is at least $k-2$. We first show that there is a SSG in which for every (α, β)-NE $\alpha > 1.0719$. This means that in every state there is a player who can reduce her contribution by at least a factor of 1.0719. In particular,

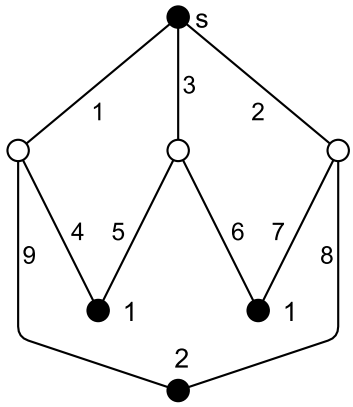


Figure 3: A SSG without a NE. Vertex labels indicate player ownership, edge labels are used for identification in the analysis given in the text.

this game has no exact NE. In addition, we provide an initial lower bound for the stability ratio of approximate NE purchasing T^*. Our bound of $\alpha > 1.1835$ is, however, significantly lower than the bound of $\alpha \geq 1.5$ shown in [4] for general connection games.

Lemma 5 *There is a SSG in which for every (α, β)-approximate Nash equilibrium $\alpha > 1.0719$. For $(\alpha, 1)$-approximate Nash equilibria the bound increases to $\alpha > 1.1835$.*

Proof. Consider the game in Figure 3. We suppose that $c(e_j) = 1$ for $j = 3, \ldots, 9$. Furthermore, we set $c(e_1) = c(e_2) = x$ and try to adjust the value x such that the minimum achievable stability ratio is maximized.
Consider any (α, β)-NE. Clearly, the network purchased must be connected and include all terminals and the source s. Once players purchase a network with cycles, they can drop edges until the network becomes a tree. In addition, they only need edges on paths between s and a terminal t. Dropping unnecessary edges from the network only decreases the payments of players and increases the cost of possible deviations. Such a transformation decreases the stability ratio. Thus, the minimum stability ratio is obtained for a state purchasing a tree network in which every leaf is a terminal. The following case analysis derives bound for the stability ratio in each class of trees including some subset of the edges $\{e_1, e_2, e_3\}$.

Case 1.1 : Suppose only e_1 is included in the tree (the case with only e_2 is symmetric). We set up inequalities to place a general upper bound on certain deviation possibilities. These upper bounds can only be strengthened if certain parts of the network are purchased and the player has to contribute less to the respective edges. $s_p(e)$ denotes the contribution of player p to edge e.

$$\begin{array}{rcll} |s_1| & \leq & \alpha(s_1(e_2) + s_1(e_7) + s_1(e_1) + s_1(e_4)) & \leq \alpha(2 + x + s_1(e_1)) \\ |s_2| & \leq & \alpha(s_2(e_1) + s_2(e_9)) & \leq \alpha(1 + s_2(e_1)) \end{array}$$

The cheapest tree in this case has cost $4 + x$, hence adding the inequalities yields the best possible bound of $4 + x \leq \alpha(3 + 2x)$, and thus $\alpha \geq 1 + \frac{1-x}{3+2x}$.

Case 1.2: Suppose only edge e_3 is bought. Any such network has cost at least 5. In this case

$$\begin{array}{rcll} |s_1| & \leq & \alpha(s_1(e_2) + s_1(e_5) + s_1(e_6)) & \leq 3\alpha \\ |s_2| & \leq & \alpha(s_2(e_1) + s_2(e_9)) & \leq \alpha(1 + x), \end{array}$$

so the stability ratio in these networks is at least $\alpha \geq 1 + \frac{1}{4+x}$. This is dominated by the bound of Case 2.2.

Case 2.1: Suppose e_1 and e_3 are purchased (the case with e_2 and e_3 is symmetric). It is possible to set up inequalities representing meaningful deviations as in Case 1.1 and to get a lower bound of $\alpha \geq 1 + \frac{1-x}{3+2x}$.

Case 2.2: Suppose e_1 and e_2 are purchased. We bound some deviations by

$$\begin{aligned} |s_1| &\leq 3\alpha \\ |s_2| &\leq \alpha(1 + s_2(e_1)) \\ |s_2| &\leq \alpha(1 + s_2(e_2)) \end{aligned}$$

It is easy to see that the minimum case is achieved by setting $y := s_2(e_1) = s_2(e_2)$. This yields

$$\begin{aligned} 2 + 2x - 2y &\leq 3\alpha \\ 1 + 2y &\leq \alpha(1 + y) \end{aligned}$$

For a stability ratio $\alpha > 1$ we need $x > 0.5$. Then the bound of Case 1.2 is dominated by the one of Case 2.1. Hence, finding the optimum x poses the optimization problem

$$\max_{x \in (0.5,1)} \max_{y \in [0,x]} \min \left(\frac{4+x}{3+2x}, \frac{2+2x-2y}{3}, \frac{1+2y}{1+y} \right).$$

In the optimum case all inner terms are equal. This results in an optimum

$$y^* = \frac{x - 3 + \sqrt{x^2 - 2x + 7}}{2}.$$

Substitution yields the problem

$$\max_{x \in (0.5,1)} \min \left(\frac{4+x}{3+2x}, \frac{5+x-\sqrt{x^2-2x+7}}{3} \right).$$

Again, the inner bounds must be equal to yield the optimum. It gives $x \approx 0.68548$ and a lower bound on $\alpha > 1.07195$.

Case 3: Finally, suppose all three e_1, e_2 and e_3 are purchased. Then the network includes three additional edges. In this case the network allows the same deviation bounds as the network of Case 2.2, but it is more costly. The minimum stability ratio is not obtained in this case.

Finally, for the case, in which T^* must be bought, let x be arbitrarily close to 1. Then T^* includes both edges e_1 and e_2 of cost x, and the corresponding bound of Case 2.2 approaches

$$\lim_{x \to 1} \frac{5 + x - \sqrt{x^2 - 2x + 7}}{3} = 2 - \sqrt{\frac{2}{3}} > 1.1835$$

This proves the lemma. □

We can embed the game of Figure 3 into the construction for the price of anarchy. The resulting game does not have a cheap NE.

Theorem 2 *The price of stability in the SSG is at least* $k - 2$.

Proof. We combine the game of Figure 3 with the game that maximizes the price of anarchy. We use $c(e_1) = c(e_2) = 0.75$ for simplicity and note that the analysis of Lemma 5 can be repeated to see that the corresponding game has no NE. Consider the resulting game depicted in Figure 4. Suppose there is a NE without the edge of cost $k-2-\epsilon$. Then players $3, \ldots, k$ must be connected with the edge of cost 1, so they all have a direct connection to the source. They do not contribute anything to the cost of the remaining edges of cost $O(\epsilon)$. Thus, the game for players 1 and 2

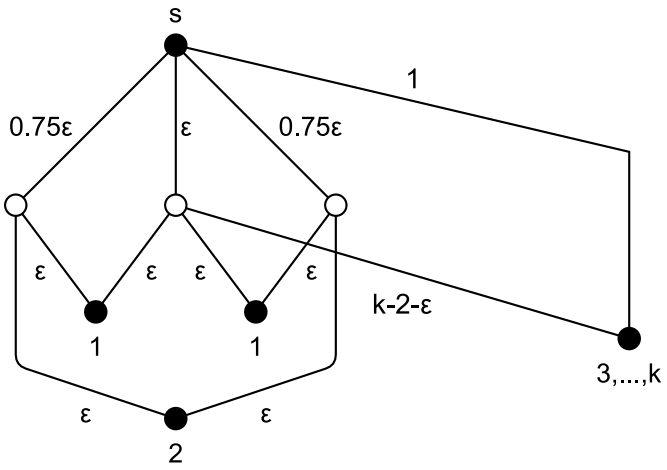


Figure 4: A SSG with price of stability arbitrarily close to $k-2$. Vertex labels indicate player ownership, edge labels indicate cost.

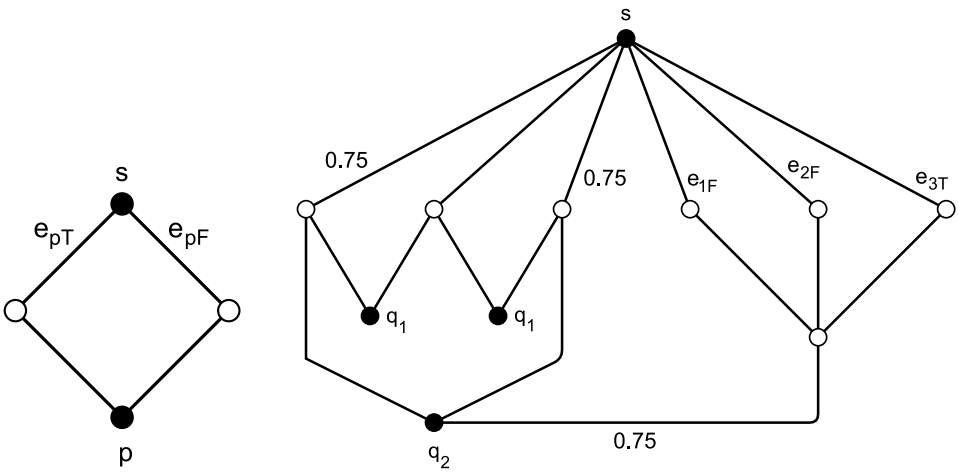
reduces to the one of Figure 3. Lemma 5 shows, however, that in this case there can be no NE. Hence, in any NE the players must purchase the edge of cost $k-2-\epsilon$. Consider the following strategies: Players $3,\ldots,k$ purchase the costly edge and the additional edge of cost ϵ to s in equal shares; player 1 purchases two edges of cost ϵ; player 2 purchases one edge of cost ϵ and one of cost 0.75ϵ. There is such a strategy combination that forms a NE. Hence, as ϵ tends to 0, the price of stability becomes arbitrarily close to $k-2$. □

Theorem 3 *In the SSG it is* NP*-hard to decide if a game has a Nash equilibrium.*

Proof. We briefly outline the proof, which uses a reduction from 3SAT. For an instance of 3SAT a SSG is constructed as follows. For each variable x_p we introduce a *variable player* p and a gadget depicted in Figure 5(a). It consists of a single terminal of p and two connections to the source s. A player has a *true* path to s including an edge e_{pT} and a *false* path including an edge e_{pF}. For each clause $\mathcal{C}_q$ we introduce two *clause players* q_1 and q_2 and a gadget depicted in Figure 5(b). It consists of a game of Figure 3 and an alternative connection to the source by a side gadget. This side gadget includes three edges e_{pT} or e_{pF}. These edges are the ones from the true or false path of the corresponding gadgets for the variables appearing in $\mathcal{C}_q$. For example, the gadget for a clause $(\overline{x_1} \vee \overline{x_2} \vee x_3)$ includes edges e_{1F}, e_{2F} and e_{3T} from the variable gadgets of variables x_1, x_2 and x_3 (c.f. Figure 5(b)). There is only one edge e_{pT} and one edge e_{pF} for each variable x_p in the whole graph.

Suppose the 3SAT instance has a satisfying assignment. Then we assign the variable players to purchase the edges of the true or false path corresponding to their assignment. This results in a total cost of 2 for each of them. Then, in each clause gadget one of the e_{pT} or e_{pF} edges is bought. For a clause $\mathcal{C}_q$ we assign player q_1 to purchase the three edges of cost 1 directly connecting her terminals to s. Player q_2 is assigned to purchase two edges of the side gadget of total cost 1.75, which connect her terminal to one edge bought by a variable player. It is easy to note that in this case no player has a possibility to connect her terminals with a lower cost by choosing a different strategy.

On the other hand suppose there is a NE. If for a clause gadget player q_2 does not use edges of the side gadget to connect her terminal, an analysis similar to the proof of Lemma 5 tells us that q_1 and q_2 do not agree upon a set of edges to purchase. Hence, player q_2 does contribute only to edges of her side gadget. In addition, this implies that no clause player contributes to the cost



(a) Gadget for a variable.

(b) Gadget for a clause ($\overline{x_1} \vee \overline{x_2} \vee x_3$).

Figure 5: Variable and clause gadgets. Vertex labels indicate player ownership. Numeric edge labels indicate cost, all unlabeled edges have cost 1.

of edges e_{pT} or e_{pF}. Thus, these edges must be fully bought by the variable players. However, to ensure a connection through the side gadgets for each clause player q_2, there must be for each clause gadget at least one edge e_{pT} or e_{pF} purchased by the variable players. As each variable player purchases exactly one of her true or false paths, this directly yields the desired satisfying assignment for the 3SAT instance.
This proves that in general deciding the existence of a NE is NP-hard. Our constructed games, however, involve only players with at most three terminals each. For such a player the optimum Steiner tree can be found in polynomial time [10], and thus it is possible to recognize a NE in polynomial time. Hence, for this special case the problem is NP-complete. □

3 Approximate Nash Equilibria

For TCGs exact NE can be hard to find or very expensive. This section studies the existence and computability of cheap approximate NE providing a trade-off between efficiency and stability. We use the concept of *connection sets* [4], which are special sets of edges of T^*.

Definition 3 *A connection set S of player p is a subset of edges of T_p, such that for each connected component C in $T^* \setminus S$ either*

1. *there is a terminal of p in C, or*
2. *any player that has a terminal in C has all of her terminals in C.*

For any Steiner vertex of degree 2 in T^* the incident edges belong to the same connection sets, so for convenience we assume that T^* has no such vertices. Suppose we assign a player to purchase exactly a single connection set. If she switches to a cheaper set of edges connecting her terminals, this keeps the purchased network feasible and improves upon the cost of T^*. Hence, the cost of a connection set for a player lower bounds any of her deviation cost. Thus, if T^* is purchased by assigning every player to pay for at most α connection sets, the state forms an $(\alpha, 1)$-NE. Note that a subset of a connection set also is a connection set.

3.1 An Algorithm for PTCGs

In connection games with two terminals per player the edges of T^* can be partitioned into equivalence classes S_Q such that e and e' belong to the same class iff $Q = \{q : e \in T_q\} = \{q : e' \in T_q\}$.

Lemma 6 *In connection games with two terminals per player each S_Q forms a connection set for all players $q \in Q$ which is maximal under the subset relation. In the PTCG connection sets S_Q form a contiguous path.*

The lemma can be proven rather easily by assuming the contrary and deriving contradictions to the definition of connection sets and tree connection requirements.
We call a connection set S_Q *needed* by Q. For the rest of this section we consider only maximal connection sets and not explicitly mention a player, as this information is given implicitly by the player subtrees the set is located in. For deriving approximate NE we again use Algorithm 1. In line 4 of Algorithm 1, however, we use a different procedure to assign the costs. For a leaf player p we pick two connection sets and assign p to purchase them. If p is not a leaf player, $s_p = 0$.
We must carefully choose the connection sets that are assigned to a leaf player p. For instance, there might be two connection sets, which are needed by player sets differing only by p. If p is assigned to purchase none of these sets, then after the player is removed in line 7 of Algorithm 1, the two connection sets will be needed by the same player set. As aruged above, however, this identifies them as one connection set. Hence, in this case two distinct connection sets would be considered as one connection set after removal of p. Naturally, this would destroy our argumentation if this connection set is assigned to another player considered in later iterations. In addition, a connection set needed only by p must be assigned to p, because otherwise it will remain unpurchased. Avoiding these problems provides candidate connection sets for the assignment to p.

Definition 4 *A connection set is called an* endangered set *for player p if*

1. *it is needed only by p. We call such a connection set a* personal set*.*
2. *it is needed by the set of players $Q \cup \{p\}$, and there is another connection set (called a* forcing set*) needed by the set Q, with $Q \neq \emptyset$ and $p \notin Q$. We call such a connection set a* community set*.*

Indeed, for any leaf player there are at most two endangered sets.

Lemma 7 *For any leaf player in a PTCG there are at most two endangered sets.*

Proof. As there is only one personal set, we must show that there is at most one community set. Assume for contradiction that for a leaf player p there are several community sets. Arbitrarily pick two distinct forcing sets S_1' and S_2' with player sets Q_1 and Q_2, respectively. The corresponding community sets are denoted S_1 and S_2. We denote $Q = Q_1 \cup Q_2$.
Consider the tree T^*. 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 p. As the subtree T_p for each player is a path, the third component (denoted C_3) contains a terminal of each player in $(Q_1 \cup Q_2) - (Q_1 \cap Q_2)$. For the first two cases we suppose there is no forcing set in C_3.

Case (a): S_1' and S_2' are located in C_1 and C_2, each in a different component. Hence, after removal of S_1' and S_2' components C_4 and C_5 evolve (see Figure 6(a)). Now all terminals of players in Q are distributed to C_3, C_4 and C_5. If the underlying graph structure allows it, we can reconnect these components into a component and C_1 and C_2 into a second component. This would yield a disconnected graph that satisfies the connection requirements. This is a contradiction to the presence of tree connection requirements, no matter whether such a connection is actually possible with the edges from the underlying graph or not.

Case (b): S_1' and S_2' are located in C_1 and C_2, both 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 Q_1. As S_1' is in C_2, all players of Q_1 need both S_1 and S_2, so $Q_1 \subset Q_2$.

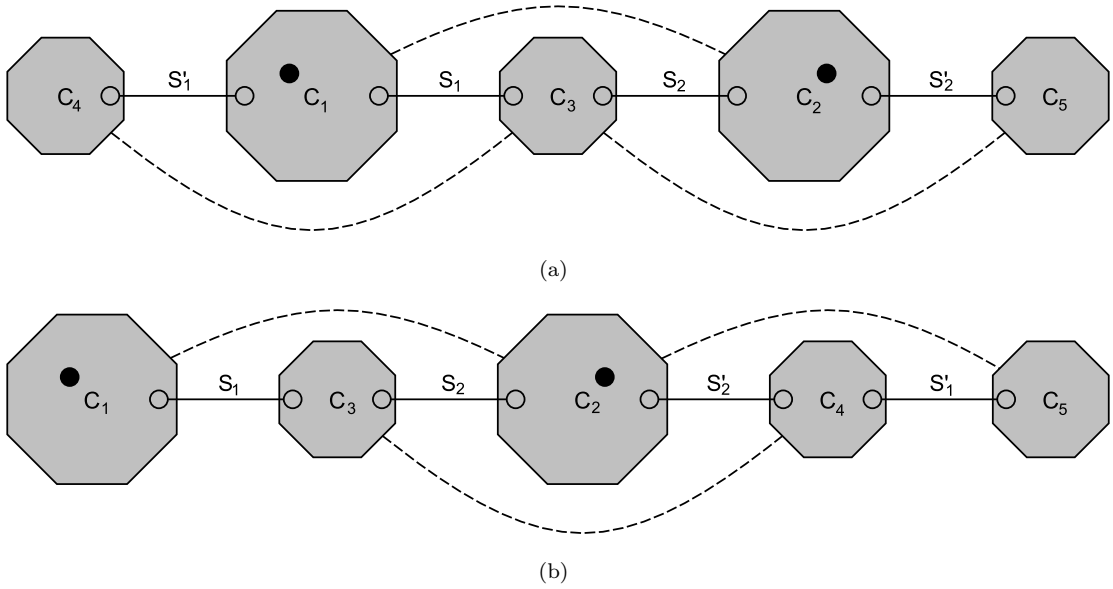


Figure 6: Component structures in the presence of more than two community sets for player p. Replacing connection sets S_1, S_2, S'_1, and S'_2 with dashed edges creates feasible, unconnected networks. Filled vertices are terminals of p.

Hence, in C_3 there is one terminal of each player of $Q_2 - Q_1$. In C_2 there is only one terminal of each player in Q_1. If we remove S'_2, we split off a new component C_4 containing one terminal of each of the players in Q_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 Q_1 (see Figure 6(b)). In C_4 one terminal of each of the players in $Q_2 - Q_1$ 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. This again violates the tree connection requirements.

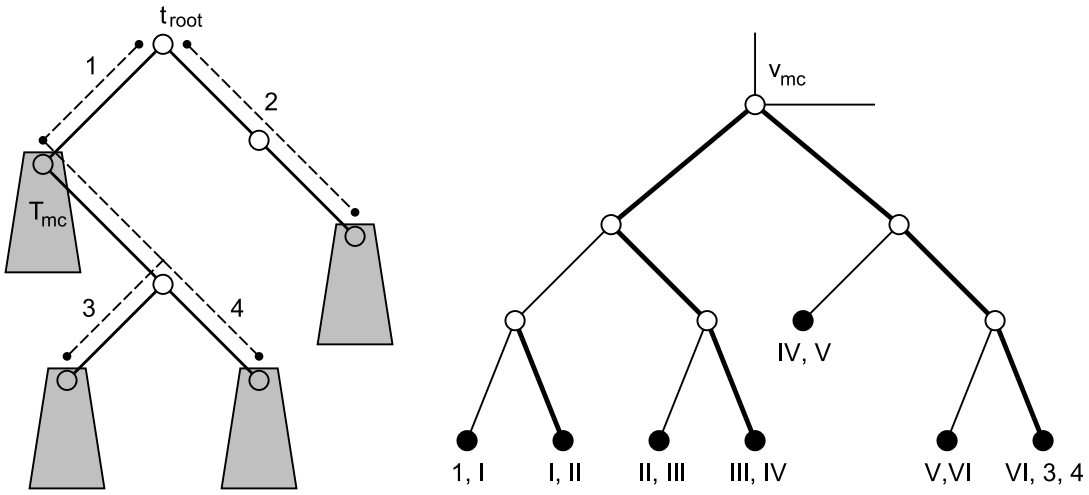
Case (c): Suppose one forcing set (w.l.o.g. S'_1) is located in C_3. This means that $Q_1 \cap Q_2 = \emptyset$. The tree requirements ensure, however, that for each pair of players $q_1 \in Q_1$ and $q_2 \in Q_2$ there is a sequence of players that transitively require a connection between q_1 and q_2. Note that p cannot be part of this sequence as she is a leaf player. In particular, this means there is at least one player whose path includes either S_1 or S'_1. This is a contradiction to the definition of community and forcing sets.

This proves that there is only one community set, which yields at most two endangered sets for a leaf player. □

In line 4 of Algorithm 1 we thus simply assign a leaf player to purchase the endangered sets. This ensures that all connection sets of T^* are assigned, and the connection sets considered and assigned in later iterations correspond to original connection sets. Then the algorithm works correctly. It can be combined with recent approximation algorithms to yield the following theorem.

Theorem 4 *For any social optimum solution T^* in a PTCG there exists a $(2,1)$-approximate Nash equilibrium such that the purchased edges are exactly T^*. A $(2+\epsilon, 1.55)$-approximate Nash equilibrium can be computed in polynomial time, for any constant $\epsilon > 0$.*

Proof. The algorithm assigning endangered connection sets is still inefficient, because it requires T^* as input. For the translation into a polynomial time algorithm we use the idea presented in [4]. It is possible to use a β-approximation algorithm for Steiner Tree to get an initial approximation T. Assume this tree is optimal and assign connection sets to a leaf player. After the



(a) (b)

Figure 7: Schematic illustration of child player distribution for a parent player p. (a) Distribution of hierarchical players 1-4. Gray trapezoids indicate contracted subtrees that are needed only by player p. (b) Detailed view of the distribution of personal players I-VI in the subtree labeled T_{mc} in (a). Bold edges indicate personal sets of personal players after the assignment of the algorithm

assignment consider each of her sets independently. In particular, for a connection set S_Q assume a cost of 0 for all $e \in T \backslash S_Q$ and calculate the shortest path in G between the terminals of p. If S_Q is not optimal, then replace it with the cheapest path and output the improved network. In this way the network T can feasibly be improved, which yields a restart of Algorithm 1 on the improved network. To ensure a polynomial number of restarts, fix parameter $\gamma = \frac{\epsilon c(T)}{(1+\epsilon)n\beta}$ in the beginning (with ϵ small enough to ensure $\gamma < \min_{e \in E} c(e)$). For checking optimality of a connection set S_Q, temporarily reduce the cost of each edge in S_Q by γ. Then a cheaper path improves the cost of the tree by at least an amount of γ. This yields at most $\frac{(1+\epsilon)n\beta}{\epsilon}$ restarts of the framework. After the algorithm has run to completion, a last post-processing step is needed to restore the original costs of the edges. Each player is assigned to pay only $c(e) - \gamma$ of the cost of each of her edges. The remaining cost of at most γ is split between all players proportionally to the total contribution of each player to the cost of the tree. By repeating the analysis of [4] this yields at most an ϵ-factor deterioration in the stability ratio. The theorem follows with the recent 1.55-approximation algorithm for Steiner Tree [18]. □

3.2 An Algorithm for TCGs

In this section we adjust the idea of assigning connection sets to get $(2,1)$-NE purchasing 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 two terminals per player. The child players have the same terminals as the parent player, and they are distributed such that the child player game is a PTCG. Then, the algorithm assigning endangered sets can be used to assign T^* such that each child player purchases at most two connection sets. The union of these connection sets yields only two connection sets for the parent player.

Theorem 5 *For any social optimum tree T^* in a TCG there exists a $(2,1)$-approximate Nash equilibrium exactly purchasing T^*.*

Proof. The following pattern is used to divide a leaf parent player p into *hierarchical* and *personal* child players. Then we process these child players such that the union of assigned connection sets forms two connection sets for p. This suffices to prove the theorem. For our division of player p it is possible to disregard all non-lonely terminals of p but one, as the corresponding connection requirements can be left for other players to satisfy. Denote this last remaining non-lonely terminal by t_{root}. If the player has only lonely terminals, we pick t_{root} arbitrarily. Then consider T^* rooted at t_{root} in BFS-order. For an edge e needed only by p, the tree connection requirements guarantee that the subtree below e is also needed only by p. Contract all such edges that are needed only by p. Denote this adjusted tree by T_{adj} and consider it again in BFS-order rooted at t_{root}. For each vertex $t \in V_p$ we introduce a new child player. She gets assigned t and the nearest ancestor vertex that is a terminal of p (see Figure 7(a)). These child players are termed *hierarchical players*.
Consider the portions of the tree that were contracted to form T_{adj}. For each maximal connected subtree $T_{mc} \subset T_p$ that is needed only by p, let v_{mc} be the root vertex that represents T_{mc} in T_{adj}. Let player q be the first hierarchical child player, who got assigned the root vertex v_{mc}. This player strives to connect upwards in T_{adj}. Now we consider T_{mc} in DFS-order and consider the first encountered terminal of p. If this is not the root v_{mc}, we relocate child player q to this terminal. For each new terminal t_x encountered in the DFS order, we introduce a new child player and assign her terminals t_{x-1} and t_x. Except for the remaining hierarchical players at v_{mc} there is only one child player with a lonely terminal in T_{mc} at all times during this assignment. Finally, consider the last terminal t in the DFS-scan of T_{mc}. We assign all hierarchical players connecting downward in T_{adj} to t instead of the root v_{mc}. Child players introduced in the DFS-scan of the components T_{mc} are called *personal players*, because they divide parts needed only by p (see Figure 7(b)).
After the division of a parent player the algorithm for PTCGs is used to assign connection sets. In any iteration a leaf child player is picked and assigned to purchase her endangered sets, however, we prefer to pick personal over hierarchical leaf players. Thus, the procedure works roughly bottom up to t_{root}. Finally, one connection set for the parent player p is formed by the union of all personal sets for the child players. The other connection set is the union of the community sets. Actually, a slightly stronger statement holds.

Lemma 8 *If the child players of a parent player p are created and eliminated in the described way, the removal of the child players' personal and community sets creates only components that contain terminals of p, respectively.*

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 personal, community, and forcing sets share a common vertex if they exist.*

Proof. If there is no forcing set, there is no community set and the lemma follows trivially. So let there be a forcing set S_Q needed by player set Q. Suppose for a leaf player p the sets do not share a vertex. Remove the community set, and let C_1, C_2 be the components with and without the lonely terminal of p, respectively.

Case (a): Suppose S_Q is in C_2 and remove it. This splits C_2 into two components. We denote by C_2' the remaining component including the terminal of p. The other component is denoted by C_3, and it contains one terminal of each player in Q. Now remove all edges that connect to the lonely terminal t_p of p in C_1. Connect all resulting components except for t to C_3. Then connect the vertex v to C_2' (see Figure 8(a)). All connection requirements are met, but there is a solution with two components. This contradicts the presence of tree connection requirements. Hence, S_Q must be in C_1.

Case (b): Suppose S_Q is in C_1 and remove it. Similar to Case 1 we refer to C_1' and C_3 as resulting components after removal of S_Q. Now suppose there is another player q with a terminal located in C_1'. T_q can only include one of S_Q and S_{Q+p}, so $q \notin Q$, and she must have both her terminals in C_1'. Again isolate the lonely terminal t_p. Then construct two components, one

consisting of t, C_2 and C_3; the other one consisting of all other components (see Figure 8(b)). This generates a feasible solution with two components, which is a contradiction of tree connection requirements. Once there are no terminals in C_1', there can be no connection sets in C_1', except for the personal set S_p needed only by p. It remains to show is that S_p must be located in C_1'.

Case (c): Suppose S_p is in C_2 and remove it. Again we denote by C_2' the component with the terminal of p and by C_3 the other one. Observe that C_3 must contain all terminals of Q. Then remove S_Q from C_1 generating C_1' and C_3'. We can isolate the components containing terminals of Q from the rest of the components (see Figure 8(c)). A feasible network with two components is possible, which contradicts the presence of tree connection requirements.

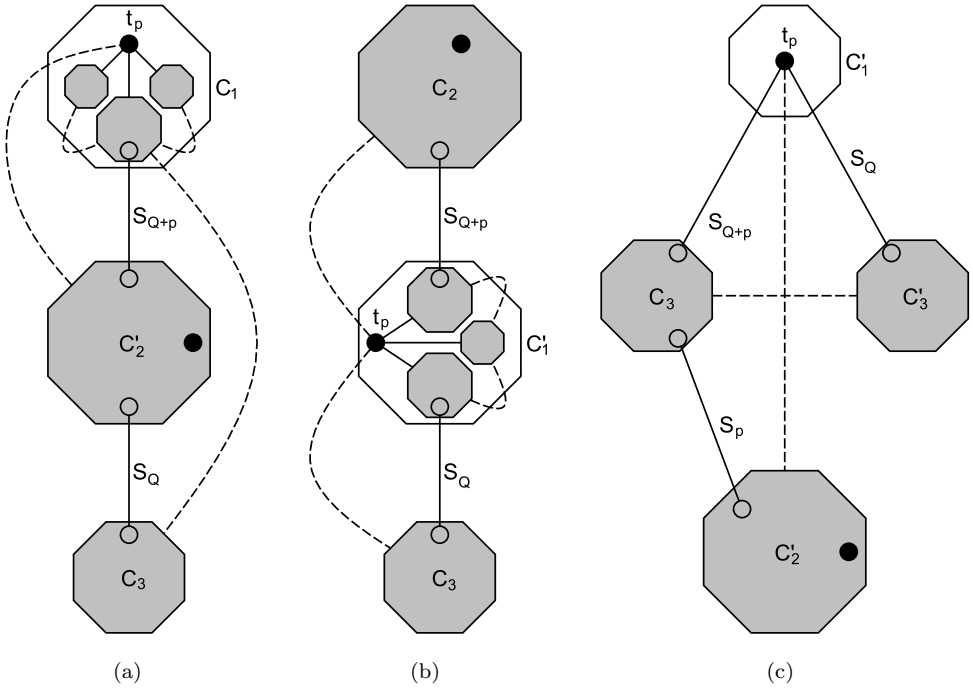


Figure 8: Component structures when endangered sets do not share a vertex. Replacing solid with dashed edges creates feasible, unconnected networks. Filled vertices are terminals of p.

If S_p exists, it is in C_1', and the three connection sets share a Steiner vertex. Otherwise, the two connection sets meet at the lonely terminal of p. This concludes the proof of Lemma 9. □

Proof. (of Lemma 8) We use an inverse induction to show the lemma. Suppose the algorithm has assigned all edges to child players. We reverse the elimination order of child players and consider player and edge additions instead of player removals and edge contractions. The child player that was eliminated last is now the one that is inserted first. It is obvious that for the first inserted (i.e. last eliminated) child player the lemma holds. This is the base case of our induction. Now suppose the property holds after we have inserted in the reverse order a given number of child players from one or more parent players and their assigned edges. Then consider the insertion of an additional child player q of a parent player p. This can be either a hierarchical or personal child player. Recall the elimination order outlined above. Player q can have a personal set. Consider the union of all personal sets of child players of p eliminated later (i.e. inserted already). By the induction hypothesis after removal of this union every evolving component contains a terminal of p. If in addition to that the personal set of q is removed, the only additional component that evolves is the lonely terminal of q. This proves the induction for the personal sets.

For the community sets the case is more complicated. Consider the hierarchical players. Their community sets are always needed by at least one additional player, which is not a child player of p. If the inserted player q is a hierarchical player, consider her terminal in the lower of the two subtrees T_{mc} containing her terminals. This lower terminal is in the component newly created by the removal of her community set. This is ensured by the hierarchical structuring and the fact that other players are also present on the community set.
If q is a personal player, we consider the subtree T_{mc} that she is located in. Recall that T_{mc} is needed only by p, and note that the root vertex v_{mc} of T_{mc} does not have to be a terminal of p. In addition, v_{mc} might be incident to community sets of hierarchical players. The complete subtree T_{mc} must be purchased by p, hence it consists completely of personal and community sets of child players of p. The lemma is proven if we can show that every vertex in T_{mc} is either a terminal of p or connected by personal sets to a terminal of p. In addition, the root v_{mc} must be connected by personal sets to the terminal of the hierarchical players connecting downward in the tree (if any). This serves to keep the above given argument for hierarchical players feasible. Now consider as an additional invariant that for every connected subtree $T' \subseteq T_{mc}$ the vertex v' closest to t_{root} is connected by personal sets to the terminal considered last during the DFS-scan of T'. Due to the DFS-based construction of personal players we always eliminate first the player constructed last in T_{mc}. Consider the subtrees q is located in. At the current time she is inserted last, so in turn of the players present she was eliminated first. Therefore, we know she was constructed last, and because of that she cannot have *only* a community set. So if q has a community set, she also has a personal set. Then, due to the properties shown in Lemma 2, there is a Steiner vertex v between these sets. Suppose now the endangered sets of q are contracted, then by construction the lonely terminal of the second-last introduced child player or a hierarchical player is joined with v. Using the induction hypothesis and the fact that v is connected with a personal set to the lonely terminal of q, we see that the invariant holds in T_{mc}. In particular, every vertex stays connected by a path of personal sets to a terminal of p, and hence no component without a terminal of p can evolve once all community sets of p are removed from T^*. For illustration see Figure 7(b), in which the bold lines indicate the personal sets of the child players I-VI. This proves the induction hypothesis for community sets of hierarchical and personal players, and Lemma 8 follows. □

The splitting for a leaf parent player p creates two edge sets, which upon removal yield only components including terminals of p. If such a set is removed, all resulting components must be reconnected to form a feasible network. Hence, these edge sets are connection sets for p. It also ensures that in our induction we can add the community (personal) sets of q to the sets of community (personal) sets of other child players of p. This completes the proof of Theorem 5. □

The next lemma ensures that the assignment of personal and community sets for a leaf parent player p does not depend on the splitting of the other parent players. In an iteration of our framework we can thus assign edge costs to a parent player p assuming an arbitrary splitting of other parent players. This ensures that the algorithm can find the correct personal and community sets for child players in polynomial time.

Lemma 10 *The endangered sets of child players of a leaf parent player* p *are independent of the division of other parent players.*

Proof. Again we use an inductive argument based on a single child player. Suppose we have a child player q of parent player p, who is removed from the game. Consider an arbitrary splitting of the other parent players into child players obeying the tree connection requirements. The personal set of q is independent of the splitting of the other parent players, which proves the lemma for the personal set.
Suppose q has a community S_c set needed by $Q \cup \{q\}$ and a forcing set S_f needed by Q. We denote by v the vertex that both sets connect to. Consider a different player p' with a child player in Q and $Q \cup \{q\}$. p' cannot have a terminal at v and $T_{p'}$ 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 p' on S_c and S_f, which would contradict the assumption that a

community set for q is present. Now consider a different splitting of p', which results in different player sets needing S_c and S_f. There must be at least one child player of p' needing each of these sets, because they are both in $T_{p'}$. However, as there is no terminal or alternate connection at v that is in $T_{p'}$, any child player of p' needing S_c also needs S_f. Hence, S_c remains the community set for q. This argument can easily be adjusted for more players. □

Procedure `ApproxNash`(T, c, p, γ) for computing $(3.1 + \epsilon, 1.55)$-NE for TCGs

Input: A feasible tree T, a cost function c, a selected player p, a constant γ
Output: A payment function s_p or a tree T^+

1 Pick a non-lonely terminal as t_{root}
2 Disregard all non-lonely terminals of p except for t_{root} from her set V_p
3 Generate T_{adj} by contracting subtrees T_{mc} only needed by p
4 Create hierarchical players on T_{adj}
5 Expand T_{adj} and create child players on each T_{mc}
6 Run algorithm for child players of p; prefer choice of personal over hierarchical leaf players
7 Assign p to purchase connection sets assigned to her child players
8 **for** each of the two connection sets S **do**
9 Create c_S by $c_S(e) = c(e) - \gamma$ for $e \in S$ and $c_S(e) = c(e)$ otherwise
10 Create G_S by contracting all edges of $T \backslash S$.
11 Run 1.55-approximation algorithm on G_S and c_S for terminals of p
12 **if** returned solution S' is cheaper than S under c_S **then**
13 **return** $T^+ = T - S + S'$
14 **return** s_p

Theorem 6 *For a TCG a* $(3.1 + \epsilon, 1.55)$*-approximate Nash equilibrium can be computed in polynomial time, for any constant* $\epsilon > 0$*.*

Proof. Procedure 3 sketches and summarizes the described steps to compute approximate NE for general TCGs. The complete algorithm again uses Algorithm 1, and in line 4 it calls Procedure 3 to assign some cost of T^* to the parent player p. Note that for TCGs we can use the improvement steps on connection sets to obtain a polynomial time algorithm using the same scaling ideas as in Theorem 4. As each connection set is now a tree, we use the 1.55-approximation algorithm [18] for Steiner Tree not only to compute a starting solution, but also to compute improvements for connection sets. This yields a stability ratio of $3.1 + \epsilon$. The theorem follows. □

3.3 A Tightness Argument

Our Procedure 3 and ADTW proposed in [4] both rely on the concept of connection sets. In this section we will argue that with respect to connection games the analytic power of connection sets is limited. In particular, algorithms that approach the problem of finding good approximate NE relying on connection sets cannot achieve a significantly lower stability ratio.
The difference between our algorithm and ADTW is that the assignment procedure used by ADTW does not employ the structural information of our child player splitting. With the structure of TCGs a splitting of parent players and a hierarchical elimination order are possible. This avoids the matching step ADTW uses to assign edge costs to players. This is crucial for achieving a guarantee of two connection sets.
Similar to our algorithm ADTW does not employ cost sharing of edges. The next theorem shows that no deterministic algorithm using only T^* as input can improve the guarantees even if it

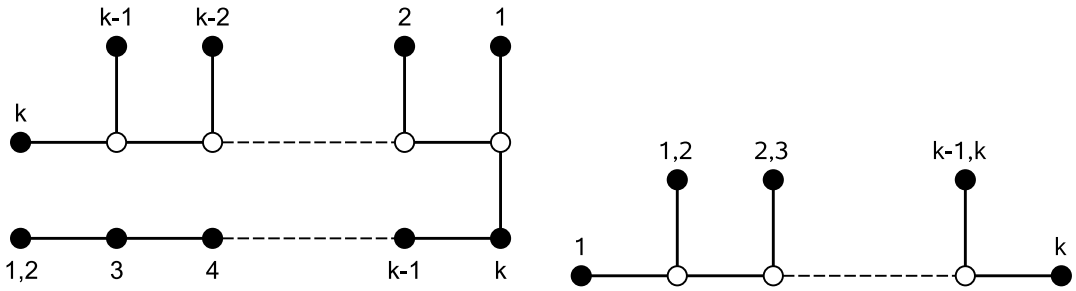


Figure 9: Optimal trees T^* for games yielding tightness of the analysis of the algorithms. One player can have a cheap additional edge of cost $1+\epsilon$ connecting her terminals. This is unknown to the algorithms and thus gives a lower bound on the stability ratio of (a) $3-\epsilon$ for general connection games and (b) $2-\epsilon$ for TCGs.

uses cost sharing. Thus, ADTW for general connection games and our algorithm for TCGs yield optimal stability ratios with respect to this class of algorithms. Our algorithm provides a better guarantee on TCGs, because ADTW can assign three connection sets to a player of a TCG with a cheap alternate path.

Theorem 7 *For any $\epsilon > 0$ there is a connection game such that any deterministic algorithm using only T^* as input constructs a state purchasing T^* with stability ratio at least $3-\epsilon$.*

Proof. Consider a game with two terminals per player and an optimal solution T^* of cost $3k-3$ shown in Figure 9(a). All edges of T^* have cost 1. There is at least one player that pays a cost of $3-\frac{3}{k}$. One player can have an alternative path of cost $(1+\epsilon)$ outside T^*. As this path is not known to the algorithm, the best approach is to equilibrate payments between players. It assigns each player p to pay a cost of $3-\frac{3}{k}$ for parts inside her path T_p. As k approaches infinity, the stability ratio becomes arbitrarily close to 3. □

Theorem 8 *For any $\epsilon > 0$ there is a PTCG such that any deterministic algorithm using only the optimum solution T^* as input constructs a state purchasing T^* with stability ratio at least $2-\epsilon$.*

Proof. An argument similar to the proof of the previous theorem for the game in Figure 9(b) shows that it is optimal to assign each player to contribute a cost of $2-\frac{1}{k}$ within her subtree. So as a deterministic algorithm working only with T^* our algorithm delivers the optimum worst-case guarantee. □

Theorem 9 *For any $\epsilon > 0$ there is a PTCG for which ADTW constructs a $(3-\epsilon, 1)$-approximate Nash equilibrium.*

Proof. Consider the game in Figure 9(b). ADTW proceeds as follows. At first each player is assigned to purchase her personal set. Afterwards, it picks two terminals of a player and assigns the path to be purchased by players that include parts of the path in their subtree. In particular, the terminals are assigned to purchase edges of the path. In the following iterations the paths to the purchasing terminals are assigned to other terminals and so on. Finally, a player is assigned to purchase all edges that were assigned to her terminals. The distribution of edges to terminals is done in a matching step. This suffices to provide a bound of 3 for the general case, and the previous Theorem 7 showed that this is optimal for an algorithm using only T^* in general connection games. For TCGs, however, it might yield an unlucky assignment. If in our example the

assignment starts by picking player 2, the matching can assign player 1 to purchase two connection sets. With the personal set this results in three connection sets. If all edges have cost 1 and there is an alternative path of cost arbitrarily close to 1, player 1 yields a stability ratio of arbitrarily close to 3. □

These tightness results cannot be easily strengthened for the polynomial time variants of the algorithms, because they heavily depend on tightness results and solution properties of the underlying approximation algorithms for Steiner Tree and Steiner Forest. It is, for instance, not known, whether the performance ratio of the recent 1.55-approximation algorithm [18] is tight.

4 Conclusion

In this work we have considered a non-cooperative game for network creation, in which selfish agents strive to build globally connected networks. Our characterization of the hardness and quality of exact Nash equilibria is reasonably tight. There is, however, potential for efficient algorithms to compute approximate NE with improved stability ratios. The concept of connection sets is a handy tool when constructing and analyzing algorithms. We have shown the boundaries of this concept by showing that using connection sets alone it is not possible to significantly improve the stability ratio over the guarantees presented for our algorithms. Nevertheless, we believe that with a different, deeper structural analysis improved ratios are possible. Additionally, the consideration of extended games based on other network design problems should lead to results with more predictive value for reality. Towards this end we proposed the *backbone game* [14], which is based on the Group Steiner tree problem. We were able to show that the price of stability is 1 in a single source scenario and to provide an algorithm for $(1+\epsilon, \beta)$-NE, in which β is bounded by a polylogarithmic function of the input size. These results, however, extensively use techniques presented here and in [4]. Further work needs to be done towards a deeper understanding of the stability properties in these scenarios.

Acknowledgement

The author would like to thank an anonymous reviewer for detailed comments that led to significant improvements in the presentation of the results.

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)*, pages 89–98, 2006.

[3] E. Anshelevich, A. Dasgupta, J. Kleinberg, T. Roughgarden, É. Tardos, and T. Wexler. 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] J. Cardinal and M. Hoefer. Selfish serive installation in networks. In *Proc 2nd Workshop Internet & Network Economics (WINE)*, 2006.

[6] 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)*, pages 99–107, 2005.

[7] H.-L. Chen and T. Roughgarden. Network design with weighted players. In *Proc 18th Symp Parallelelism in Algorithms and Architectures (SPAA)*, pages 29–38, 2006.

[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] E. Demaine, H. Mahini M.T. Hajiaghayi, and M. Zadimoghaddam. The price of anarchy in network creation games. In *Proc 26th Ann ACM Symp Principles of Distributed Comp (PODC)*, 2007.

[10] S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1:195–207, 1972.

[11] 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.

[12] M. Goemams and D. Williamson. A general approximation technique for constrained forest problems. *SIAM J Comp*, 24(2):296–317, 1995.

[13] M. Hoefer. Non-cooperative facility location and covering games. In *Proc 17th Intl Symp Algorithms and Computation (ISAAC 2006)*, 2006.

[14] M. Hoefer. Non-cooperative tree creation. In *Proc 31st Symp Math Foundations of Comp Sci (MFCS)*, pages 517–527, 2006.

[15] 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.

[16] 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.

[17] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proc 16th Ann Symp Theoretical Aspects Comp Sci (STACS)*, pages 404–413, 1999.

[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.