Prize Collecting Bottleneck Steiner Problems: A Combinatorial Approach

Fabian Hargesheimer *

Abstract

We design a purely combinatorial 2-approximation algorithm for the Prize Collecting Bottleneck k-STP, which arises naturally in several network design problems. Our algorithm is optimal in the sense that it matches the lower bound on approximability of the non-prize collecting variant of the problem. We also consider the case of an unbounded number of Steiner nodes and show exact solutions for the Penalty- and Quota Bottleneck Steiner Tree Problem. Finally, we introduce a more powerful model in the form of Prize Collecting Steiner Forest where prizes are vectors, representing networks where nodes have different values to different subnets. We design extensions for our previous algorithms yielding exact solutions for both Penalty- and Quota Bottleneck Steiner Forests using this extended model.

1 Introduction

The *Steiner Tree Problem*(STP) on graphs is the problem of finding a minimum interconnection of a set of vertices, called terminals, with the possibility of using optional vertices, called Steiner nodes, to decrease the length of the connection. The NP-Completeness of this problem has been shown in [1, 2]. The currently best known lower bound on the approximability of that problem is 1.01063 [3], while the best known algorithm regarding approximation ratio is the LP-based approach by Byrka et al. with an approximation ratio of $1.39 + \epsilon$ for every $\epsilon > 0$ [4]. In this paper, we will only cover network design problems on graphs, and as such, all following results and statements are made in that context.

*Dept. of Computer Science, University of Bonn. Work supported by Hausdorff Doctoral Fellowship.
Email: hgh@cs.uni-bonn.de

The *Prize Collecting Steiner Tree Problem* is a variant of the STP where the absolute requirement of connecting all terminals is relaxed to connecting only those which lead to an improvement of the modified objective function. Prize collecting problems where first introduced by Balas [5], defining a variant of the Traveling Salesman Problem where nodes can be skipped for the resulting tour, imposing a penalty on the objective function. An additional quota constraint defines a minimum fraction of prizes that has to be collected by visiting their nodes. For the prize collecting variant of the STP, the best known algorithm with an approximation ratio of 1.9672 was shown by Archer et al. [6] in 2009. It is based on the primal-dual algorithm for the same problem by Goemans and Williamson [7]. Like most published algorithms for prize collecting problems, they only include the penalty mechanic, but not the quota constraint.

In *Bottleneck Steiner Tree Problems*, the objective is to minimize the cost of the most expensive edge used in the solution. The inverse of the Bottleneck STP, finding a Steiner tree that maximizes cost of the least expensive contributing edge, is called *Maximum Scatter Steiner Tree* and can be solved by solving the Bottleneck STP on the same instance. The Bottleneck STP can be solved exactly in polynomial time [11]. The *Bottleneck k-STP*, which places a restriction on the number of Steiner nodes that can be visited in the solution, was shown NP-hard to approximate within a factor of $2 - \epsilon$ by Abu-Affash, Carmi and Katz in 2011. They also presented a 2-approximate algorithm for this problem in the same paper [12].

Note that the name k-STP is also frequently used to refer to the problem of finding a minimum Steiner tree connecting at least k terminals, a problem related to the k-Minimum Spanning Tree (k-MST) [8, 9, 10]. In the context of this paper, k-STP will always refer to the STP with a restriction on the number of Steiner nodes.

In this paper we design approximation algorithms for prize collecting variants of the Bottleneck STP and k-STP, which model naturally occurring network design problems, e.g. design of wireless networks, pipelining and other networks with weakest/strongest link criteria under economical constraints. Note that in applications, we are often interested in the reverse question: Given a certain bottleneck value, what is the minimum number of Steiner nodes required to connect the network. This problem can be solved easily by using our algorithm in a binary search. Interestingly, unlike most known algorithms for prize collecting network design problems, our algorithms reach the approximability lower bound for the non-prize collecting variants, i.e. we have proven the problems do not become computationally harder by including prize collecting mechanics. A related problem, the Prize Collecting Bottleneck TSP, was covered by us earlier [13].

2 2-Approximation for Penalty Bottleneck k-STP

We first consider the most common type of prize collecting problem, the penalty problem. In *Penalty Problems* we are allowed to skip nodes, but for every unvisited in the solution, we incur a penalty equal to the prize assigned to that node, i.e. our objective function becomes minimizing the edge costs plus the penalties for unvisited nodes. In the case of bottleneck problems, this means minimizing the maximum value of (1.) the most expensive edge and (2.) the highest incurred penalty. The problem of minimizing the maximum of (1.) the most expensive edge and (2.) the sum of all penalties incurred is called penalty-sum problem instead, and will not be covered here. The algorithm proposed in this section can be thought of as an extension of the Bottleneck k-STP algorithm presented in [12].

The main idea behind our algorithm is the following: Since the bottleneck clearly has to be either the cost of an edge, the penalty for not visiting a vertex, or both, we iterate through all possible bottleneck values in the form of an event structure, while simultaneously keeping track of two graphs to check if we can achieve a connecting network: One representing the currently utilizable edges in regards to edge costs and one representing the current set of terminals (in prize collecting bottleneck problems, the notion of terminals must take into account the current bottleneck value) and the costs of connecting them in regard of number of Steiner nodes required. We terminate if a Steiner tree fulfilling all requirements (edge costs, penalties, number of Steiner points) can be constructed.

Let $G = (V, E, c, \pi)$ be an graph with vertices V, edges $E \subseteq \mathcal{P}_2(V)$ and associated functions $c : E \to \mathbb{R}_+$, assigning costs to edges and $\pi : V \to \mathbb{R}_+$, assigning to each vertex a prize, which will be incurred as penalty if the node is not visited in the solution. Let G fulfil triangle inequality. We call $v \in V$ Steiner node if $\pi(v) = 0$. Let k be the number of Steiner nodes allowed to be used to construct a solution.

We define an event structure X as a list containing all edges and vertices sorted by non-decreasing values of their costs and prizes respectively, and we call single elements of the list events. The active element of this list tracks the current bottleneck value. We denote elements of X representing edges by x^e and elements representing vertices by x^v.

We will use two additional data structures, each containing an undirected graph: $G_i = (V, E_i, c)$ contains the subgraph of G so that $c(e) \leq x_i, \forall e \in E_i$ while we define $G_> = (V_>, E_>, w)$ as a complete graph over all vertices $\pi(v) > x_i, \forall v \; in \; V_>$, i.e. the complete graph over all remaining terminals for a given event. The edge costs in $G_>$ are

given by the cost function w, which we will describe below.

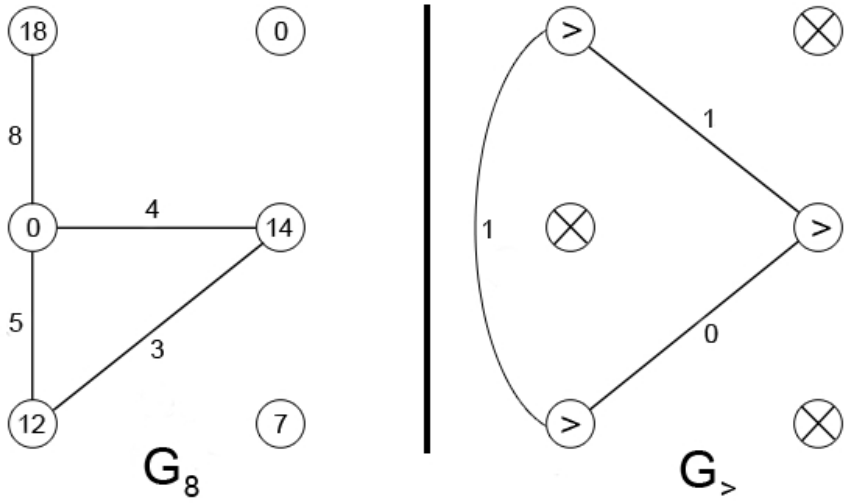

Figure 1: Example of G_i and $G_>$ after reaching an event with bottleneck value 8. The edges of $G_>$ show the weights w induced by the distance of nodes in G_i measured in Steiner nodes.

We use $\delta_i(p, q)$ to denote the path with the minimal number of Steiner points between vertices p and q in G_i and define the cost function w on $G_>$ as $w(p, q) =$ number of Steiner points in $\delta_i(p, q)$. Note that the induced graph $G_>$ is not a metric: The weight function can assign a value of zero to edges $e_>$. We denote by T the minimum spanning tree in $G_>$ under weight function w.

In $\boxed{1}$ we construct our event structure as defined above. We then remove any x^e occurring after the last x^v. Together with $\boxed{5}$, $\boxed{6}$ this deals with the special case where the empty solution, i.e. no edge taken, highest penalty of all vertices paid, is the optimal solution. The pruning step is necessary to make sure the algorithm does not terminate with a suboptimal solution. When terminating in the case of a non empty solution, the type of the resulting bottleneck (vertex or edge) is determined by the type of the last processed event. If there are events for edges and vertices with the same value as the bottleneck value, the resulting Steiner tree can also have bottlenecks in both an edge and a not connected node.

The updating steps for our support structures are represented by $\boxed{9}$, $\boxed{10}$: If x' is an event representing an edge, G_i is updated by adding all edges $e \in E, c_e = i$ to E_i.If x' is an event representing a vertex, $G_>$ is updated by removing all vertices $v \in V_>, \pi_v = i$ and all edges adjacent to one of the removed vertices. We then update the costs w for

Algorithm 1: P-BN-k-ST($G = (V, E, c, \pi), k$)

```
1:  Construct and prune event structure X
2:  C(T) ← ∞
3:  while C(T) > k do
4:    x' ← select next element of X
5:    if x' is last element of X
6:      return ∅-solution
7:    else
8:      i ← value of x'
9:      if x' ∈ x^v then G_> = (V_>, E_>)
10:     if x' ∈ x^e then G_i = (V, E_i)
11:     for each edge (p, q) ∈ E_> do
12:       w(p, q) ← number of Steiner points in (δ_i(p, q))
13:     T ← MST of G_> under w
14:     C(T) ← Σ_{e∈T} ⌊w(e)/2⌋
15: Construct k-Steiner tree using G, T, G_i
```

all $e \in E_>$ and construct a minimum spanning tree T in $G_>$ in 11 to 13. Whether we choose to count vertices $v \in V : 0 < \pi(v) < i$ against the limit of Steiner points and thus our calculation for w, depends on our modeling goal. In a mobile radio network for example, every receiver is also a sender, so we could argue that those vertices become *free* Steiner nodes.

In 14 we set $C(T)$ to $\sum_{e \in T} \lfloor w(e)/2 \rfloor$, and 3 stops our iteration over the events of X if $C(T) \leq k$. Using Lemma 3 from [14] we know that for any spanning tree T' of $G_>$, the following holds: $C(T) \leq C'(T)$. From Lemma 3.1 in [12] we know that if G_i contains a k-Steiner tree, then $C(T) \leq k$. Thus, our algorithm stops at the event with the lowest bottleneck value at which we can construct a Steiner tree with at most k Steiner vertices.

Finally, our resulting Steiner tree is constructed in 15: For each edge $e = (p, q) \in T$, we construct a path from p to q in G using at most $\lfloor \frac{w(e)}{2} \rfloor$ of the Steiner points in $\delta_i(p, q)$.

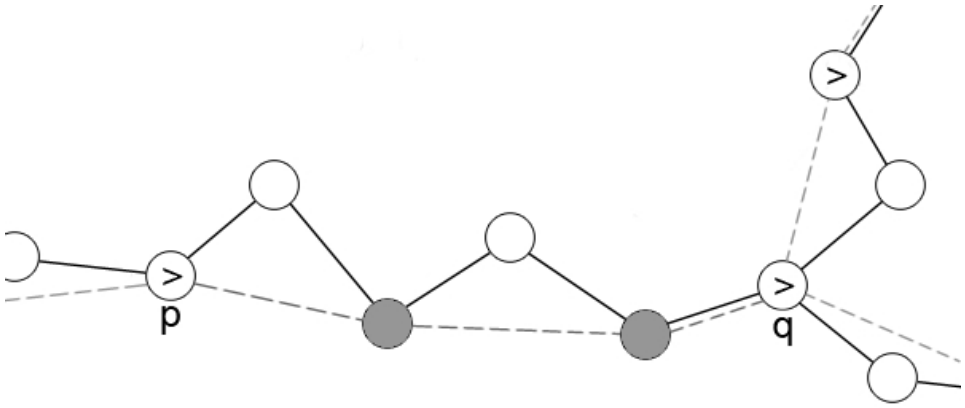

Figure 2: Construction of of the resulting Steiner tree: Since triangle inequality holds for G, for any path (p,q), a path taking at most half of the Steiner nodes of the original path can be constructed with at most twice its bottleneck value.

We will now proof our main theorem:

Theorem 2.1.
There exists a polynomial-time algorithm solving the Penalty Bottleneck k-Steiner Tree Problem with a worst-case approximation ratio of 2.

Proof: Algorithm 1 can terminate only in two ways: It either reaches the final element of X, and the empty solution requiring no Steiner points is chosen, or a Steiner tree with less then k Steiner nodes is constructed. In the latter case, since G is a metric, using the above construction of the resulting Steiner tree clearly results in a bottleneck value of at most twice the value of the most expensive edge in G_i, independent of whether the last processed event was increasing the edge cost threshold or removing terminals from $G_>$. If the empty solution is chosen, the resulting bottleneck value is exact, as can be seen easily. Thus the worst-case approximation ratio is 2. □

3 Unrestricted Steiner Problems

We will now look at the case of an unrestricted number of Steiner nodes, i.e. the Prize Collecting Bottleneck STP. As mentioned before, for the non prize collecting variant, an exact solution is computable in polynomial time [11], as such it is of interest whether this holds for the prize collecting variant as well. We will consider both penalty and

quota versions of the Bottleneck STP and show that this is indeed the case, even when the underlying graph does not fulfill triangle inequality.

3.1 Penalty Bottleneck STP

For the Penalty Bottleneck Steiner Tree, an exact solution can be obtained by a simplification of Algorithm 1:

Algorithm 2: P-BN-ST($G = (V, E, c, \pi)$)

```
1:  Construct and prune event structure X
2:  T ← ∅
3:  while T = ∅ do
4:    x' ← select next element of X
5:    if x' is last element of X
6:      return ∅-solution
7:    else
8:      i ← value of x'
9:      if x' ∈ x^v then G_> = (V_>, E_>)
10:     if x' ∈ x^e then G_i = (V, E_i)
11:     for each edge (p, q) ∈ E_> do
12:       w(p, q) ← 1 if (δ_i(p, q)) < ∞, ∞ else
13:     T ← Spanning Tree of G_>
14: Construct Steiner tree using G, T, G_i
```

Here we iterate over the events in X until we can construct a spanning tree in $G_>$. Since we only care about the bottleneck value, the total length of T, as well as the length of the resulting Steiner tree do not matter. Since we are not bounded by the number of Steiner nodes, the construction of the resulting Steiner tree in $\boxed{14}$ will be done simply by taking the path $\delta_i(p, q)$ for each edge $e = (p, q) \in T$ in G. Since we iterate over all possible bottleneck values sorted by non-decreasing value, it is easy to see that the algorithm finds the exact value of the bottleneck. We can thus summarize this subsection with the following theorem:

Theorem 3.1.
There exists an exact-solution algorithm solving the Penalty Bottleneck Steiner Tree problem in polynomial time.

3.2 Quota Bottleneck STP

Quota Problems are a different type of prize collecting problem. The quota criterion was used in combination with the penalty criterion in the first paper on prize collecting problems [5], but since then these two have been treated mostly as seperate problems [6, 7, 15, 16]. In Quota Problems, the objective function becomes minimizing the cost of edges taken (minimizing the most expensive edge taken in bottleneck problems), with the additional constraint that the sum of prizes in the solution has to be at least a certain fraction Q of the sum of all available prizes.

The idea behind this algorithm is, again, to iterate over all possible bottleneck values. Since we do not include the penalty criterion, this time the bottleneck clearly has to be the value of an edge. By using a list of edge values in non-decreasing order as events, we can incrementally construct a subgraph containing connected components which form possible solutions, and test those components for feasibility.

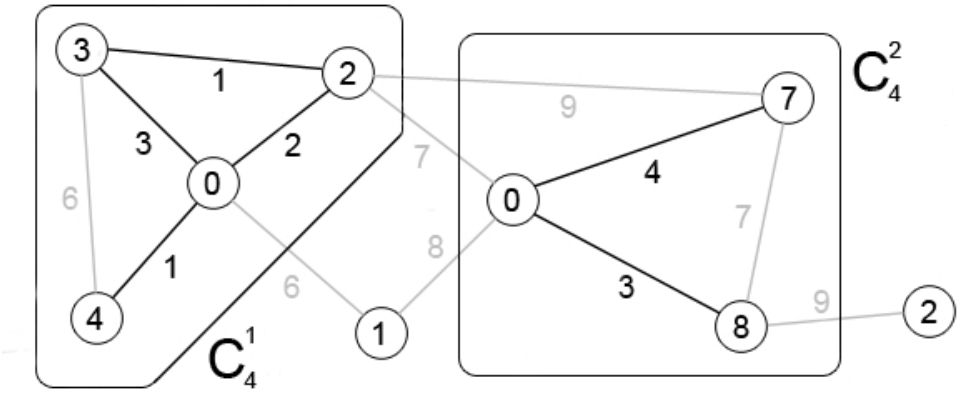

Figure 3: An example for Algorithm 3. For a required quota of 0.5, a component reaching that quota is found at a bottleneck threshold of 4.

Let $G = (V, E, c, \pi)$ be a connected, undirected graph with vertices V, edges E, a function $c : E \to \mathbb{R}_+$ assigning costs to edges and a function $\pi : V \to \mathbb{R}_+$ that assigns to each vertex a prize. We call $v \in V$ Steiner node if $\pi(v) = 0$.

We denote by X a list containing all edges sorted by non-decreasing values of their cost. We call any $x \in X$ an event. The active element of this list tracks the current bottleneck value. We denote by $G_i = (V, E_i, c, \pi)$ the subgraph of G so that $c(e) \leq x_i, \forall e \in E_i$. For a given G_i we will denote its j-th connected components by C_i^j. We define $Q \in [0, 1]$ as the required quota and denote by T a spanning tree in the first

component C_i^j so that its spanned vertices have a total prize value fulfilling the quota requirement.

Algorithm 3: Q-BN-ST($G = (V, E, c, \pi), Q$)

```
1:  Construct event structure X
2:  repeat
3:      x' ← select next element of X
4:      i ← value of x'
5:      G_i = (V, E_i)
6:      for each connected component C_i^j ∈ G_i do
7:          if (Σ_{v∈C_i^j} π(v)) / (Σ_{v∈V} π(v)) > Q
8:              T ← Spanning Tree of C_i^j
9:              return T
10:         else continue
```

Since we know that at least one feasible solution exists in the form of the bottleneck being equal to the most expensive edge in G, represented by the last element of X, we know that the algorithm will terminate. It is easy to see that the non-decreasing ordering of X will result in finding the lowest possible bottleneck value. Thus we can summarize this section with the following theorem:

Theorem 3.2.
There exists an exact-solution algorithm solving the Quota Bottleneck Steiner Tree Problem in polynomial time.

4 Steiner Forest Problems

The *Steiner Forest Problem* is a generalization of the Steiner Tree Problem with multiple distinct sets of terminals. All terminals within each of the sets have to be interconnected, while the sets themselves can be connected, but do not have to be. The best currently known upper bound algorithm with an approximation ratio of 2 was given by Jain [17]. For the penalty version of this problem, an upper bound of 2.54 was shown by Sharma et al. in 2007 [18].

In the non prize collecting version of this problem, the sets of terminals (called subnets from here on) are typically disjunct, which bounds the number of such sets by the number

of vertices in the graph. When using prize collecting variants of this problem to model economical problems, it can however make sense to use a generalization where prizes are vectors, representing the different value a vertex has for different subnets. For the same reason, we will also allow different subnets to have different quota requirements. Using this more general definition for the Prize Collecting Steiner Forest Problem, the number of possible subnets is no longer bounded. In order to achieve polynomial running time for our algorithms, we will only look at instances where the number of subnets is at most polynomial. We will now show that under this restriction, both Penalty Bottleneck Steiner Forest and Quota Bottleneck Steiner Forest are solvable in polynomial time with exact solutions, even if the underlying graphs do not fulfill triangle inequality.

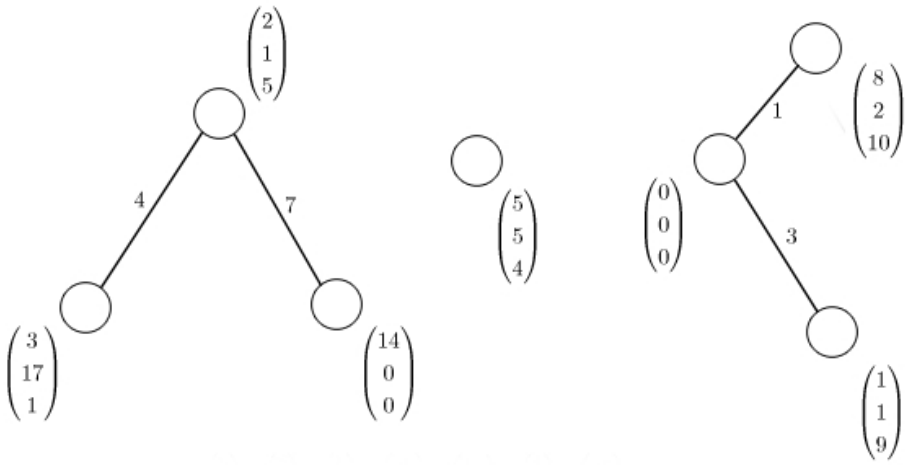

Figure 4: Example of a Penalty Bottleneck Steiner Forest with prize vectors. Here, the bottleneck value of 8 is caused by the penalty for not connecting the upper right node to the other nodes of subnet 1.

4.1 Penalty Bottleneck Steiner Forests

The algorithm presented here is a natural extension of algorithm 2, but in this case we terminate only when our objective is reached for every subnet of G. We will only discuss the differences to the tree algorithm.

To assign different prizes for vertices for each of the m subnets, we define the prize function as $\pi : V \to \mathbb{R}_+^m$. Our event structure X will be a list containing all edges and all elements of each prize vector, sorted by non-decreasing values. For $j = 1, .., m$ we denote by $G_>^j$ the complete graph over all vertices whose j-th prize value is above the

current bottleneck value, and by T^j a spanning tree covering $G^j_>$. We denote the Steiner tree for the j-th subnet in G by H^j and the resulting Steiner forest by F.

Algorithm 4: P-BN-SF($G = (V, E, c, \pi), m$)

Construct and prune event structure X
$T^1 \leftarrow \emptyset$
while $T^j = \emptyset$ for any j= 1,..,m **do**
  $x' \leftarrow$ select next element of X
  if x' is last element of X
    return $\emptyset$-solution
  else
    $i \leftarrow$ value of x'
    if $x' \in x^e$ **then** $G_i = (V, E_i)$
    for j= 1,..,m **do**
      if $x' \in x^v$ **then** $G^j_> = (V^j_>, E^j_>)$
      for each edge $(p, q) \in E^j_>$ **do**
        $w^j(p, q) \leftarrow 1$ if $(\delta_i(p, q)) < \infty, \infty$ *else*
      $T^j \leftarrow$ *Spanning Tree* of $G^j_>$
for j= 1,..,m **do**
  Construct Steiner tree H^j using G, T^j, G_i
Construct F using $\cup_{1,..,j,..,m} H^j$

The construction of our solution F is done by taking the union of all Steiner trees H and then replacing all multi-edges by single edges and removing cycles by removing random edges. Since the values of X are non-decreasing, it is easy to see that the bottleneck value can not decrease by these operations, since otherwise the algorithm would have terminated at that lower value. Combining this with the arguments for the tree algorithm, this yields the following theorem:

Theorem 4.1.
For a polynomially bounded number of subnets, there exists a polynomial time exact-solution algorithm solving the Penalty Bottleneck Steiner Forest Problem.

4.2 Quota Bottleneck Steiner Forest

The idea behind this extension to algorithm 3 is analogue to the extension used in the previous algorithm, so we will only cover it for completions sake.

In addition to allowing different prizes for vertices for each subnet, we will also allow each of the m subnets to require a different quota, denoted by the vector Q^m.

Algorithm 5: Q-BN-SF($G = (V, E, c, \pi), \vec{Q}, m$)

1: Construct event structure X
2: $T^1 \leftarrow \emptyset$
3: **while** $T^n = \emptyset$ for any n= 1,..,m **do**
4: $x' \leftarrow$ select next element of X
5: $i \leftarrow$ value of x'
6: $G_i = (V, E_i)$
7: **for** n= 1,..,m **do**
8: **for** each connected component $C_i^j \in G_i$ **do**
9: **if** $\frac{\sum_{v \in C_i^j} \pi^n(v)}{\sum_{v \in V} \pi^n(v)} > Q^n$
10: $T^n \leftarrow$ *Spanning Tree* of C_i^j
11: Construct F using $\cup_{1,..,n,..,m} T^n$

Since the resulting forest is constructed from the trees only by removing edges that are redundant for connectivity, the sum of prizes collected is not changed by this step, and all quotas are still met. Otherwise the same arguments as in the previous algorithms apply, leading us to our final theorem:

Theorem 4.2.
For a polynomially bounded number of subnets, there exists a polynomial time exact-solution algorithm solving the Quota Bottleneck Steiner Forest Problem.

5 Further Research

In this paper, we designed a 2-approximation algorithm for the Penalty Bottleneck k-STP and exact-solution algorithms for the Penalty- and Quota Bottleneck STP with no restriction on the number of Steiner nodes, as well as for their associated forest problems. For the k-restricted cases of Quota Bottleneck STP and both forest problems,

the question of approximability is still open. The same is true for the prize collecting versions of these problems when applying both the penalty and the quota criterion simultaniously, as proposed by Balas initial definition of prize collecting problems [5].

The result on the Penalty Bottleneck k-STP raises hope that there may exist other prize collecting problem instances reaching the same approximation ratio as their non-prize collecting variants. Prize collecting variants of other bottleneck problems in particular, like the aforementioned Penalty Bottleneck TSP (see also [13]), might be candidates for tight combinatorial approximation algorithms, due to their similar structure.

References

[1] M.R. Garey, R.L. Graham, D.S. Johnson. The complexity of computing Steiner minimal trees. *SIAM Journal of Applied Mathematics*, 32(4):835-859, 1977.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy. Proof verification and hardness of approximation problems. *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS '92)*, 14-23, 1992.

[3] M. Chlebik, J. Chlebikova. Approximation hardness of the Steiner tree problem on graphs. *SWAT 2002: Proceedings of the 8th Scandinavian Workshop on Algorithm Theory*, 170-179, Springer Verlag, 2002.

[4] J. Byrka, F. Grandoni, T. Rotvohs, L. Sanita. An Improved LP-based Approximation for Steiner Tree. *STOC 2010*, 583-592, 2010.

[5] E. Balas. The prize collecting travelling salesman problem. *Networks*, 19:621-636, 1989.

[6] A. Archer, M. Bateni, M. Hajiaghayi, H. Karloff. Improved Approximation Algorithms for Prize-Collecting Steiner Tree and TSP. *Proc. of the 50th Annual Symp. on Foundations of Computer Science*, 40(2):309-332, 2009.

[7] M. Goemans, D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296-317, 1995.

[8] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, S. S. Ravi. Spanning Trees Short or Small. *SIAM Journal of Discrete Mathematics*, 9:178-200, 1996.

[9] S. Arora, G. Karakostas. A 2+ϵ approximation for the k-MST Problem. *Proc. of the 11th Annual SIAM Symp. on Discrete Algorithms*, 754-759, 2000.

[10] N. Garg. Saving an epsilon: A 2-approximation for the k-MST Problem in graphs. *Proc. STOC'05*, 394-402, 2005.

[11] C. W. Duin, A. Volgenant. The partial sum criterion for Steiner trees in graphs and shortest paths. *European Journal of Operations Research*, 97:172-182, 1997.

[12] A. Karim Abu-Affash, P. Carmi, M. J. Katz. Bottleneck Steiner Tree with Bounded Number of Steiner Vertices. *23rd Canadian Conference on Computational Geometry*, 39-43, 2011.

[13] F. Hargesheimer. A Note on the Prize Collecting Bottleneck TSP and Related Problems. *CS-Report*, 85336:1-9, University of Bonn, 2013.

[14] L. Wang, D. Z. Du. Approximations for a bottleneck Steiner problem. *Algorithmica*, 32:554-561, 2002.

[15] B. Awerbuch, Y. Azar, A. Blum, S. Vempala. New approximation guarantees for minimum-weight k-trees and prize-collecting salesman. *SIAM Journal of Computing* 28(1):254-262, 1999.

[16] G. Ausiello, V. Bonifaci, S. Leonardi, A. Marchetti-Spaccamela. Prize Collecting Traveling Salesman and Related Problems. *Handbook of Approximation Algorithms and Metaheuristics* 40:1-13, Chapman and Hall/CRC, 2007.

[17] K. Jain. Factor 2 for Generalized Steiner Network. *Combinatorica* 21(2):39-60, 2001.

[18] Y. Sharma, C. Swamy, D. P. Williamson. Approximation algorithms for prize-collecting forest problems with submodular penalty functions. *SODA '07 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithm* 1275-1284, 2007.