22 Steiner Forest

We will obtain a factor 2 approximation algorithm for the Steiner forest problem by enhancing the primal–dual schema with the idea of growing duals in a synchronized manner. The Steiner forest problem generalizes the metric Steiner tree problem, for which a factor 2 algorithm was presented in Chapter 3. Recall, however, that we had postponed giving the lower bounding method behind that algorithm; we will clarify this as well.

As in the Steiner tree problem (Theorem 3.2), the main case of the Steiner forest problem is also the metric case (see Exercise 22.2). However, the primal–dual algorithm remains the same for both cases, so we don't impose this restriction.

Problem 22.1 (Steiner forest) Given an undirected graph $G = (V, E)$, a cost function on edges $c : E \to \mathbf{Q}^+$, and a collection of disjoint subsets of V, $S_1, \ldots S_k$, find a minimum cost subgraph in which each pair of vertices belonging to the same set S_i is connected.

Let us restate the problem; this will also help generalize it later. Define a *connectivity requirement function* r that maps unordered pairs of vertices to $\{0, 1\}$ as follows:

$$r(u, v) = \begin{cases} 1 \text{ if } u \text{ and } v \text{ belong to the same set } S_i \\ 0 \text{ otherwise} \end{cases}$$

Now, the problem is to find a minimum cost subgraph F that contains a u–v path for each pair (u, v) with $r(u, v) = 1$. In general, the solution will be a forest.

22.1 LP-relaxation and dual

In order to give an integer programming formulation for this problem, let us define a function on all cuts in G, $f : 2^V \to \{0, 1\}$, which specifies the minimum number of edges that must cross each cut in any feasible solution.

$$f(S) = \begin{cases} 1 \text{ if } \exists\ u \in S \text{ and } v \in \overline{S} \text{ such that } r(u, v) = 1 \\ 0 \text{ otherwise} \end{cases}$$

Let us also introduce a 0/1 variable x_e for each edge $e \in E$; x_e will be set to 1 iff e is picked in the subgraph. The integer program is:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e & & (22.1) \\ \text{subject to} \quad & \sum_{e:\ e \in \delta(S)} x_e \geq f(S), & S \subseteq V \\ & x_e \in \{0,1\}, & e \in E \end{aligned}$$

where $\delta(S)$ denotes the set of edges crossing the cut $(S, \overline{S})$.

Following is the LP-relaxation of (22.1); once again, we have dropped the redundant conditions $x_e \leq 1$.

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e & & (22.2) \\ \text{subject to} \quad & \sum_{e:\ e \in \delta(S)} x_e \geq f(S), & S \subseteq V \\ & x_e \geq 0, & e \in E \end{aligned}$$

The dual program is:

$$\begin{aligned} \text{maximize} \quad & \sum_{S \subseteq V} f(S) \cdot y_S & & (22.3) \\ \text{subject to} \quad & \sum_{S:\ e \in \delta(S)} y_S \leq c_e, & e \in E \\ & y_S \geq 0, & S \subseteq V \end{aligned}$$

Notice that the primal and dual programs form a covering and packing pair of LPs (see Section 13.1 for definitions).

22.2 Primal–dual schema with synchronization

We will introduce a new idea in the primal–dual schema for approximation algorithms, setting it apart from the way this schema is used for designing exact algorithms. The later algorithms work *on demand* – in each iteration, we pick one unsatisfied complementary slackness condition, and satisfy it by modifying the primal and dual solutions suitably. The new idea is that of raising duals in a *synchronized manner*. The algorithm is not trying to rectify a *specific* condition. Instead, it tries many possibilities simultaneously, one of which leads to primal improvement.

Some figurative terminology will help describe the algorithm more easily. Let us say that edge e *feels* dual y_S if $y_S > 0$ and $e \in \delta(S)$. Say that set S has been *raised* in a dual solution if $y_S > 0$. Clearly, raising S or $\overline{S}$ has the same effect. Sometimes we will also say that we have raised the cut $(S, \overline{S})$. Further, there is no advantage in raising set S with $f(S) = 0$, since this does not contribute to the dual objective function. Thus, we may assume that such cuts are never raised. Say that edge e is *tight* if the total amount of dual it feels equals its cost. The dual program is trying to maximize the sum of the dual variables y_S subject to the condition that no edge feels more dual than its cost, i.e., no edge is *overtight*.

Next, let us state the primal and relaxed dual complementary slackness conditions. The algorithm will pick edges integrally only. Define the *degree of set* S to be the number of picked edges crossing the cut $(S, \overline{S})$.

Primal conditions: For each $e \in E$, $x_e \neq 0 \Rightarrow \sum_{i:\ e \in \delta(S)} y_S = c_e$.
Equivalently, *every picked edge must be tight.*

Relaxed dual conditions: The following relaxation of the dual conditions would have led to a factor 2 algorithm: for each $S \subseteq V, y_S \neq 0 \Rightarrow \sum_{e:\ e \in \delta(S)} x_e \leq 2 \cdot f(S)$, i.e., every raised cut has degree at most 2. However, we do not know how to ensure this condition. Interestingly enough, we can still obtain a factor 2 algorithm – by relaxing this condition further! Raised sets will be allowed to have high degree; however, we will ensure that *on average, raised duals have degree at most 2.* The exact definition of "on average" will be given later.

The algorithm starts with null primal and dual solutions. In the spirit of the primal–dual schema, the current primal solution indicates which cuts need to be raised, and in turn, the current dual solution indicates which edge needs to be picked. Thus, the algorithm iteratively improves the feasibility of the primal, and the optimality of the dual, until a feasible primal is obtained.

Let us describe what happens in an iteration. At any point, the picked edges form a forest. Say that set S is *unsatisfied* if $f(S) = 1$, but there is no picked edge crossing the cut $(S, \overline{S})$. Set S is said to be *active* if it is a minimal (w.r.t. inclusion) unsatisfied set in the current iteration. Clearly, if the currently picked primal solution is infeasible, there must an unsatisfied set and therefore an active set w.r.t. it.

Lemma 22.2 *Set S is active iff it is a connected component in the currently picked forest and $f(S) = 1$.*

Proof: Let S be an active set. Now, S cannot contain part of a connected component because otherwise there will already be a picked edge in the cut $(S, \overline{S})$. Thus, S is a union of connected components. Since $f(S) = 1$, there is a vertex $u \in S$ and $v \in \overline{S}$ such that $r(u, v) = 1$. Let S' be the connected component containing u. Clearly, S' is also unsatisfied, and by the minimality of S, $S = S'$. □

By the characterization of active sets given in Lemma 22.2, it is easy to find all active sets in the current iteration. The dual variables of these sets are raised in a synchronized manner, until some edge goes tight. Any one of the newly tight edges is picked, and the current iteration terminates.

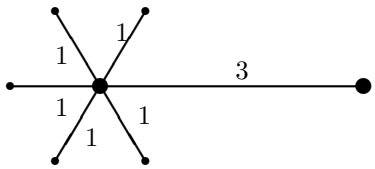
When a primal feasible solution is found, say F, the edge augmentation step terminates. However, F may contain redundant edges, which need to be pruned for achieving the desired approximation factor; this is illustrated in Example 22.4. Formally, edge $e \in F$ is said to be *redundant* if $F - \{e\}$ is also a feasible solution. All redundant edges can be dropped simultaneously from F. Equivalently, only nonredundant edges are retained.

This algorithm is presented below. We leave its efficient implementation as an exercise.

Algorithm 22.3 (Steiner forest)

1. **(Initialization)** $F \leftarrow \emptyset$; for each $S \subseteq V$, $y_S \leftarrow 0$.
2. **(Edge augmentation)** while there exists an unsatisfied set do:
   simultaneously raise y_S for each active set S, until some edge e goes tight;
   $F \leftarrow F \cup \{e\}$.
3. **(Pruning)** return $F' = \{e \in F |\ F - \{e\}$ is primal infeasible$\}$

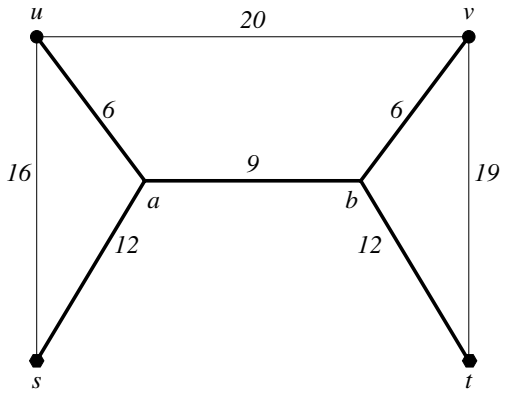
Example 22.4 Consider a star in which all edges have cost 1, except one edge whose cost is 3.



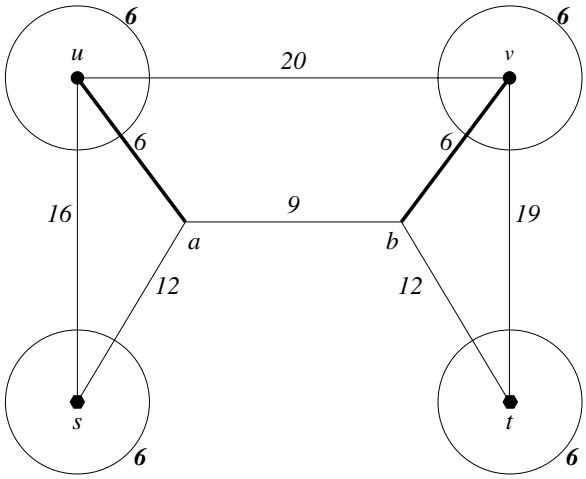
The only requirement is to connect the end vertices of the edge of cost 3. The algorithm will add to F all edges of cost 1 before adding the edge of cost 3. Clearly, at this point, F is not within twice the optimal. However, this will be corrected in the pruning step when all edges of cost 1 will be removed. □

Let us run the algorithm on a nontrivial example to illustrate its finer points.

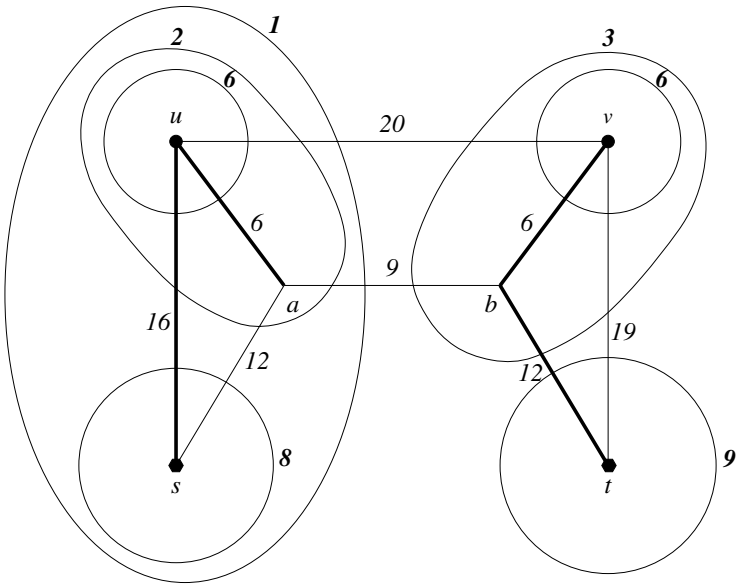
Example 22.5 Consider the following graph. Costs of edges are marked, and the only nonzero connectivity requirements are $r(u, v) = 1$ and $r(s, t) = 1$. The thick edges indicate an optimal solution of cost 45.



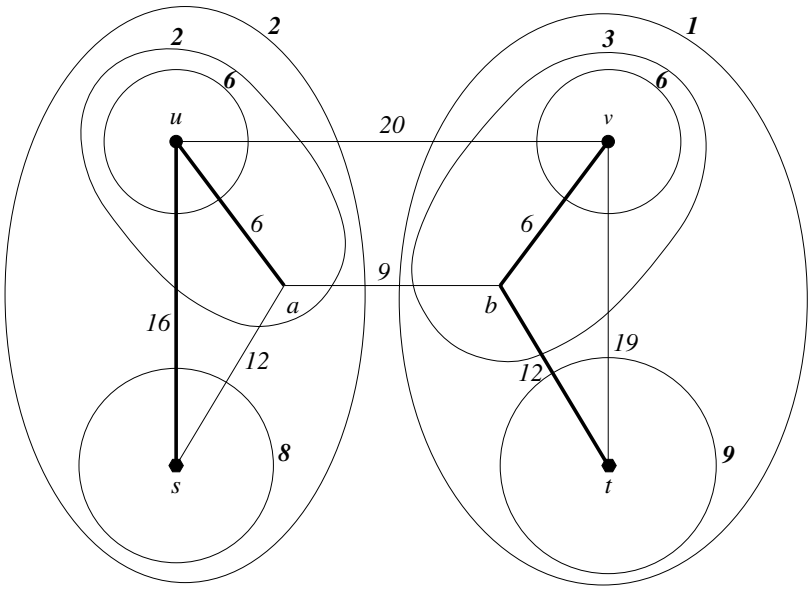
In the first iteration, the following four singleton sets are active: $\{s\}$, $\{t\}$, $\{u\}$, and $\{v\}$. When their dual variables are raised to 6 each, edges (u, a) and (v, b) go tight. One of them, say (u, a) is picked, and the iteration ends. In the second iteration, $\{u, a\}$ replaces $\{u\}$ as an active set. However, in this iteration there is no need to raise duals, since there is already a tight edge, (v, b). This edge is picked, and the iteration terminates. The primal and dual solutions at this point are shown below, with picked edges marked thick:



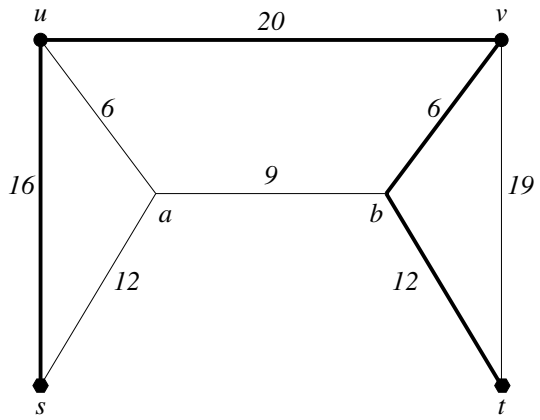
In the third iteration, $\{v, b\}$ replaces $\{v\}$ as an active set. When the active sets are raised by 2 each, edge (u, s) goes tight and is picked. In the fourth iteration, the active sets are $\{u, s, a\}, \{v\}$ and $\{t\}$. When they are raised by 1 each, edge (b, t) goes tight and is picked. The situation now is:



In the fifth iteration, the active sets are $\{a, s, u\}$ and $\{b, v, t\}$. When they are raised by 1 each, (u, v) goes tight, and we now have a primal feasible solution:



In the pruning step, edge (u, a) is deleted, and we obtain the following solution of cost 54:



□

22.3 Analysis

In Lemma 22.6 we will show that *simultaneously* deleting all redundant edges still leaves us with a primal feasible solution, i.e., it is never the case that two edges e and f are both redundant individually, but on deletion of e, f becomes nonredundant.

Lemma 22.6 *At the end of the algorithm, F' and $\boldsymbol{y}$ are primal and dual feasible solutions, respectively.*

Proof: At the end of Step 2, F satisfies all connectivity requirements. In each iteration, dual variables of connected components only are raised. Therefore, no edge running within the same component can go tight, and so F is acyclic, i.e., it is a forest. Therefore, if $r(u,v) = 1$, there is a *unique* u–v path in F. Thus, each edge on this path in nonredundant and is not deleted in Step 3. Hence, F' is primal feasible.

When an edge goes tight, the current iteration ends and active sets are redefined. Therefore, no edge is overtightened. Hence, $\boldsymbol{y}$ is dual feasible. □

Let $\deg_{F'}(S)$ denote the number of edges of F' crossing the cut $(S, \overline{S})$. The characterization of degrees of satisfied components established in the next lemma will be crucial in proving the approximation guarantee of the algorithm.

Lemma 22.7 *Consider any iteration of the algorithm, and let C be a component w.r.t. the currently picked edges. If $f(C) = 0$ then $\deg_{F'}(C) \neq 1$.*

Proof: Suppose $\deg_{F'}(C) = 1$, and let e be the unique edge of F' crossing the cut $(C, \overline{C})$. Since e is nonredundant (every edge in F' is nonredundant),

there is a pair of vertices, say u, v, such that $r(u,v) = 1$ and e lies on the unique u–v path in F'. Since this path crosses the cut $(C, \overline{C})$ exactly once, one of these vertices must lie in C and the other in $\overline{C}$. Now, since $r(u,v) = 1$, we get that $f(C) = 1$, thus leading to a contradiction. □

Lemma 22.8 $$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S$$

Proof: Since every picked edge is tight,

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \left(\sum_{S:\ e \in \delta(S)} y_S \right).$$

Changing the order of summation we get:

$$\sum_{e \in F'} c_e = \sum_{S \subseteq V} \left(\sum_{e \in \delta(S) \cap F'} y_S \right) = \sum_{S \subseteq V} \deg_{F'}(S) \cdot y_S.$$

Thus, we need to show that

$$\sum_{S \subseteq V} \deg_{F'}(S) \cdot y_S \leq 2 \sum_{S \subseteq V} y_S. \tag{22.4}$$

We will prove the following stronger claim. In each iteration, the increase in the left-hand side of inequality (22.4) is bounded by the increase in the right-hand side. Consider an iteration, and let Δ be the extent to which active sets were raised in this iteration. Then, we need to show:

$$\Delta \times \left(\sum_{S \text{ active}} \deg_{F'}(S) \right) \leq 2\Delta \times (\#\text{ of active sets}).$$

Notice that the degree w.r.t. F' of any active set S is due to edges that will be picked *during or after* the current iteration. Let us rewrite this inequality as follows:

$$\frac{\sum_{S \text{ active}} \deg_{F'}(S)}{\#\text{ of active sets}} \leq 2. \tag{22.5}$$

Thus, we need to show that in this iteration, the average degree of active sets w.r.t. F' is at most 2. The mechanics of the argument lies in the fact that in a tree, or in general in a forest, the average degree of vertices is at most 2.

Let H be a graph on vertex set V and edge set F'. Consider the set of connected components w.r.t. F at the beginning of the current iteration. In H, shrink the set of vertices of each of these components to a single node to obtain graph H' (we will call the vertices of H' nodes for clarity). Notice that in going from H to H', all edges picked in F before the current iteration have been shrunk. Clearly, the degree of a node in H' is equal to the degree of the corresponding set in H. Let us say that a node of H' corresponding to an active component is an *active node*; any other node will be called *inactive*. Each active node of H' has nonzero degree (since there must be an edge incident to it to satisfy its requirement), and H' is a forest. Now, remove all isolated nodes from H'. The remaining graph is a forest with average degree at most 2. By Lemma 22.7 the degree of each inactive node in this graph is at least 2, i.e., the forest has no inactive leaves. Hence, the average degree of active nodes is at most 2. □

Observe that the proof given above is essentially a charging argument: for each active node of degree greater than 2, there must be correspondingly many active nodes of degree 1, i.e., leaves, in the forest. The exact manner in which the dual conditions have been relaxed must also be clear now: in each iteration, the duals being raised have average degree at most 2. Lemmas 22.6 and 22.8 give:

Theorem 22.9 *Algorithm 22.3 achieves an approximation guarantee of factor 2 for the Steiner forest problem.*

The tight example given for the metric Steiner tree problem, Example 3.4, is also a tight example for this algorithm. Algorithm 22.3 places an upper bound of 2 on the integrality gap of LP-relaxation (22.2) for the Steiner forest problem. Example 22.10 places a lower bound of (essentially) 2 on this LP, even if restricted to the minimum spanning tree problem.

Let us run Algorithm 22.3 on an instance of the metric Steiner tree problem. If the edge costs satisfy the strict triangle inequality, i.e., for any three vertices u, v, w, $c(u, v) < c(u, w) + c(v, w)$, then it is easy to see that the algorithm will find a minimum spanning tree on the required vertices, i.e., it is essentially the algorithm for the metric Steiner tree problem presented in Chapter 3. Even if the triangle inequality is not strictly satisfied, the cost of the solution found is the same as the cost of an MST. Furthermore, if among multiple tight edges, the algorithm always prefers picking edges running between required vertices, it will find an MST. This clarifies the lower bound on which that algorithm was based.

The MST problem is a further special case: every pair of vertices need to be connected. Observe that when run on such an instance, Algorithm 22.3 essentially executes Kruskal's algorithm, i.e., in each iteration, it picks the cheapest edge running between two connected components. Hence it finds an optimal MST. However, as shown in Example 22.10, the dual found may be as small as half the primal.

Example 22.10 Consider a cycle on n vertices, with all edges of cost 1. The cost of an optimal MST is $n-1$. the dual found is $n/2$. Algorithm 22.3 finds a dual of value $n/2$: 1/2 around each vertex. Indeed, this is an optimal dual solution, since there is a fractional primal solution of the same value: pick each edge to the extent of half. This places a lower bound of (essentially) 2 on the integrality gap of LP (22.2), even if restricted to the minimum spanning tree problem. □

22.4 Exercises

22.1 Show, using the max-flow min-cut theorem, that a subgraph of G has all the required paths iff it does not violate any of the cut requirements in IP (22.1). Use this fact to show that IP (22.1) is an integer programming formulation for the Steiner forest problem.

22.2 Show that there is an approximation-factor-preserving reduction from the Steiner forest problem to the metric Steiner forest problem.

Show that there is no loss of generality in requiring that the edge costs satisfy the triangle inequality for the Steiner network problem.
Hint: The reasoning is the same as that for the Steiner tree problem.

22.3 How does the feasibility and approximation guarantee of the solution found change if

1. the pruning step of Algorithm 22.3 is replaced with the reverse delete step of Algorithm 18.4.
2. the reverse delete step of Algorithm 18.4 is replaced by the pruning step of Algorithm 22.3.

22.4 Give an example for which some cut raised by Algorithm 22.3 has degree at least 3 w.r.t. the primal solution found.

22.5 Run Algorithm 22.3 on an instance of the minimum spanning tree problem. Pick an arbitrary vertex as the root, and throw away all raised duals containing this vertex. Show that the cost of the tree found is twice the sum of the remaining duals.
Hint: Show that in an iteration which starts with k connected components, and lasts for time Δ, the total increase to the left-hand side of inequality (22.4) is precisely $2(k-1)\Delta$.

22.6 Let us think of running Step 2 of Algorithm 22.3 continuously in time. Thus, in unit time, a dual grows a unit amount. Consider an instance of the