Chapter 8

A General Approximation Technique for Constrained Forest Problems

This chapter studies an approximation algorithm due to M.X.Goemans and D.P.Williamson for so-called constrained forest problems, see [3] and [4]. The algorithm is based on the primal-dual method for solving a linear programming relaxation of the problem. For further details see these papers. In the context of network design problems, such approximation techniques were first presented by Agrawal, Klein and Ravi [1].

Subsequently this primal-dual method of Goemans and Williamson has been extended to several other network problems. Williamson et al [11] apply it to the generalized Steiner network problem, and Ravi and Williamson [9] apply it to the min-cost k-node connected spanning subgraph problem. Chudak et al [2] give 2-approximation algorithms based on the primal-dual method for the minimum feedback vertex set problem in undirected graphs. The primal-dual method appears to provide a versatile paradigm for the design and analysis of approximation algorithms.

We mention that Jain [7] has recently given a 2-approximation algorithm for the generalized Steiner network problem; this algorithm is based on repeatedly solving LP relaxations of the problem, but does not use the primal-dual method.

8.1 Constrained forest problems

Several problems in network design can be modeled as constrained forest problems. Some examples are: the minimum spanning tree problem, the shortest s-t path problem, the minimum-cost T-join problem, the Steiner minimal tree problem, the generalized Steiner forest problem, and the point-to-point connection problem (the last two are defined below). Moreover, assuming that the edge costs satisfy the triangle inequality, the GW (Goemans-Williamson) approximation algorithm can be applied to find approximately optimal solutions to the minimum-cost perfect matching problem and the exact tree (or cycle or path) partitioning problem. The key result in this chapter is that for (proper) constrained forest problems, the GW algorithm finds a solution whose cost is guaranteed to be within a factor of two of the optimal cost. Thus a variety of NP-hard problems in network design can be approximately solved within a factor of two, using this algorithm. The algorithm finds optimal solutions to the minimum spanning tree problem and the shortest s-t path problem;

for the first problem, the GW algorithm specializes to Kruskal's algorithm, and for the second it resembles a variant of Dijkstra's algorithm.

The GW algorithm may be viewed in several ways: It is a primal-dual algorithm, based on the linear programming relaxation of an integer programming formulation of a constrained forest problem; the algorithm is guided by the complementary slackness conditions and alternately performs updates on the primal LP solution and the dual LP solution. Also, the algorithm may be regarded as an adaptive greedy algorithm that repeatedly selects an edge with minimum reduced cost; the algorithm is adaptive, since the reduced costs are updated throughout the execution. The algorithm runs in time $O(n^2 \log n)$ on a graph with n nodes.

The definition of a proper function is given later, along with the requirements for a set of edges to satisfy a proper function.

Given an undirected graph $G = (V, E)$, nonnegative edge costs $c : E \rightarrow \Re_+$, and a proper function $f : 2^V \rightarrow \{0, 1\}$ (that is, f assigns a value of 0 or 1 to each subset S of V), the problem is to find an edge set $H \subseteq E$ that satisfies f such that its cost, $c(H) = \sum_{e \in H} c_e$, is minimum. Recall our notation $\delta(S)$, where S is a node subset: $\delta(S)$ is the set of edges with one end in S and the other end in $V \backslash S$. An edge set H is said to *satisfy* f if

$$|H \cap \delta(S)| \geq f(S), \qquad \forall S \subset V.$$

That is, H satisfies f if the number of edges of H with exactly one end in S is at least $f(S)$, for all node subsets S.

An integer linear programming formulation of a constrained forest problem is easily obtained from the above problem statement:

$$\boxed{\begin{array}{llcclll} \text{(IP)} & \text{minimize} & \sum\limits_{e \in E} c_e & x_e & & & \\ & \text{subject to} & \sum\limits_{e \in \delta(S)} & x_e & \geq & f(S), & \forall S \subset V \\ & & & x_e & \in & \{0,1\}, & \forall e \in E. \end{array}}$$

8.2 Proper functions

A function $f : 2^V \rightarrow \{0, 1\}$ (assume that $f(V) = f(\emptyset) = 0$) is called *proper* if it satisfies the two properties of symmetry and maximality (also called disjointness):

[symmetry]	$f(S) = f(V \backslash S)$,	$\forall S \subset V$,
[maximality]	$f(A \cup B) \leq f(A) + f(B)$,	for any two disjoint subsets A, B of V.

The maximality property may be restated as :

$$f(A) = 0,\ f(B) = 0 \implies f(A \cup B) = 0 \quad \text{for any two disjoint subsets } A, B \text{ of } V.$$

An example of a proper function is given in Table 8.1; the next proposition shows that the function in the example is proper.

Proposition 8.1 *Given a graph $G = (V, E)$ and a set of "terminal nodes" $N \subseteq V$, the function $f(S)$ given by*

$$f(S) = \begin{cases} 1 & \text{if } \emptyset \neq S \cap N \neq N \\ 0 & \text{otherwise} \end{cases}$$

is a proper function.

Proof: We must show that f satisfies

(i) symmetry:
For any $S \subseteq V$, if $S \cap N \neq \emptyset$ and $S \cap N \neq N$, then $(V \backslash S) \cap N \neq \emptyset$ and $(V \backslash S) \cap N \neq N$. Hence, $f(S) = f(V \backslash S)$.

(ii) maximality:
Consider disjoint subsets A and B of V, and suppose that $f(A \cup B) = 1$. Then there is a terminal node $z \in A \cup B$, say, z is in A, and also a terminal node z' that is *not* in $A \cup B$. Now consider $A \cap N$. Since $z \in A$, $A \cap N \neq \emptyset$. Since $z' \notin A \cup B$, $z' \notin A \cap N$, and so $A \cap N \neq N$. Hence, $f(A) = 1$.

∎

S	$f(S)$	S	$f(S)$
$\emptyset$	0	$\{a,b,c\}$	0
$\{a\}$	1	$\{a,b,d\}$	1
$\{b\}$	1	$\{a,b,e\}$	1
$\{c\}$	1	$\{a,c,d\}$	1
$\{d\}$	0	$\{a,c,e\}$	1
$\{e\}$	0	$\{a,d,e\}$	1
$\{a,b\}$	1	$\{b,c,d\}$	1
$\{a,c\}$	1	$\{b,c,e\}$	1
$\{a,d\}$	1	$\{b,d,e\}$	1
$\{a,e\}$	1	$\{c,d,e\}$	1
$\{b,c\}$	1	$\{a,b,c,d\}$	0
$\{b,d\}$	1	$\{a,b,c,e\}$	0
$\{b,e\}$	1	$\{a,b,d,e\}$	1
$\{c,d\}$	1	$\{a,c,d,e\}$	1
$\{c,e\}$	1	$\{b,c,d,e\}$	1
$\{d,e\}$	0	$\{a,b,c,d,e\}$	0

Table 8.1: An example of a proper function. The graph $G = (V, E)$ has 5 nodes a, b, c, d and e, and has $N = \{a, b, c\}$ as a set of "terminal nodes". The function f is given by $f(S) = 1$ if $\emptyset \neq S \cap N \neq N$, and $f(S) = 0$ otherwise

8.3 Using proper functions to model problems in network design

Four problems in network design are given here. The first three are NP-hard, whereas for the fourth problem, there is a polynomial-time algorithm based on matching theory. Each of the four problems can be modeled as a constrained forest problem such that the function f is proper. Hence, by applying the GW algorithm, an approximately optimal solution whose cost is within a factor of two of the optimal value can be obtained. For each of the four problems, first, the problem is defined, then the appropriate function f is given.

(i) Steiner tree problem:
Given a set $N \subseteq V$ of terminal nodes, find a minimum-cost tree that connects all nodes in N.

$$\forall S \subset V: \qquad f(S) = \begin{cases} 1 & \text{if } \emptyset \neq S \cap N \neq N \\ 0 & \text{otherwise.} \end{cases}$$

(ii) Generalized Steiner forest problem :
Given p sets of terminal nodes $N_1, N_2, \ldots, N_p$ (these need not be disjoint), find a minimum-cost forest that connects all nodes in $N_i, i = 1, 2, \ldots, p$. (The number of trees in the forest may be either $1, 2, \ldots,$ or p.)

$$\forall S \subset V: \qquad f(S) = \begin{cases} 1 \quad \text{if} & \begin{cases} \emptyset \neq S \cap N_1 \neq N_1 \text{ or} \\ \emptyset \neq S \cap N_2 \neq N_2 \text{ or} \\ \cdots \\ \emptyset \neq S \cap N_p \neq N_p \end{cases} \\ 0 \quad \text{otherwise.} \end{cases}$$

(iii) Point-to-point connection problem :
Given a set $C = \{c_1, \ldots, c_p\}$ of source nodes, and a set $D = \{d_1, \ldots d_p\}$ of destination nodes, find a minimum-cost forest such that each connected component of the forest contains the same number of source nodes and destination nodes.

$$\forall S \subset V: \qquad f(S) = \begin{cases} 1 & \text{if} \qquad |S \cap C| \neq |S \cap D|, \\ 0 & \text{otherwise.} \end{cases}$$

(iv) T-join problem :
Given a node set T of even cardinality, find a set of edges F of minimum cost such that every node in T is incident to an odd number of edges of F.

$$\forall S \subset V: \qquad f(S) = \begin{cases} 1 & \text{if} \qquad |S \cap T| \text{ is odd,} \\ 0 & \text{otherwise.} \end{cases}$$

The function f in part (i) has been shown already to be proper; the other three parts are left as an exercise.

8.4 The LP relaxation of (IP)

The linear programming (LP) relaxation of our integer linear programming formulation of a constrained forest problem (IP) and the dual LP are as follows. In the LP relaxation, the constraints

$$x_e \leq 1, \qquad \forall e \in E,$$

are redundant, since for every feasible solution x^*, any $x_e^* > 1$ may be replaced by $x_e^* = 1$ without violating any constraints, and moreover the objective value $\sum c_e x_e^*$ does not increase.

(LP relaxation)		(Dual LP)	
minimize	$z = \sum_{e \in E} c_e x_e$	maximize	$w = \sum_{S \subset V} f(S) y_S$
subject to	$\sum_{e \in \delta(S)} x_e \geq f(S), \quad \forall S \subset V$	subject to	$\sum_{S: e \in \delta(S)} y_S \leq c_e, \quad \forall e \in E$
	$x_e \geq 0, \quad \forall e \in E$		$y_S \geq 0, \quad \forall S \subset V$

The complementary slackness conditions are as follows:

$$\text{[primal]} \quad x_e > 0 \;\Rightarrow\; \sum_{S: e \in \delta(S)} y_S = c_e, \quad \forall e \in E$$

$$\text{[dual]} \quad y_S > 0 \;\Rightarrow\; \sum_{e \in \delta(S)} x_e = f(S), \quad \forall S \subset V$$

The GW algorithm is driven by the primal complementary slackness conditions; the dual complementary slackness conditions do not always hold, but they are needed in the analysis of the performance guarantee.

8.5 The GW algorithm for the constrained forests problem

The GW algorithm augments a subset F of the edges by one edge in each iteration (initially, $F = \emptyset$), until F satisfies the proper function f. Let F^* denote F at the end of the last iteration. Although F^* satisfies f, it may not be a minimal edge set satisfying f. The last step of the algorithm is a *clean-up* step that constructs a subset F' of F^* consisting of edges that are necessary to satisfy f. That is, F' is obtained from F^* by removing edges e such that $F^* \backslash e$ satisfies f. The clean-up step is essential for obtaining the performance guarantee.

The algorithm starts with the dual feasible solution $y_S = 0, \;\; \forall S \subset V$, and a primal solution that is infeasible, $x_e = 0, \;\; \forall e \in E$. At each iteration, corresponding to the edge set F there is a solution (that may not be feasible) of the primal LP:

$$x_e = \begin{cases} 1 & \text{if } e \in F \\ 0 & \text{otherwise.} \end{cases}$$

At any step, let C be the node set of a connected component of the subgraph formed by the current edge set F, i.e., C induces a maximal connected subgraph of (V, F). Then C is called an *active component* if $f(C) = 1$. Note that if $C \subset V$ is an active component, then the current edge set F does *not* satisfy

$$|\delta(C) \cap F| \geq f(C),$$

but for every proper subset $C' \subset C$ the current edge set F does satisfy

$$|\delta(C') \cap F| \geq 1 \geq f(C').$$

At each iteration, the algorithm finds the maximum ϵ such that for each active component C, the dual variable y_C can be increased by ϵ without violating any constraint of the dual LP, and then increases y_C for all active components C by ϵ. After this increase, one or more edges $e^* \in \delta(C)$, C an active component (clearly, $e^* \in E \backslash F$), satisfy

$$\sum_{S \subset V : e^* \in \delta(S)} y_S = c_{e^*},$$

i.e., the edge e^* becomes "tight" in the sense that the corresponding constraint in the dual LP now holds with equality. One such edge $e^* = ij$ is added to F, that is, in the primal LP, x_{e^*} is increased from zero to one. Clearly, e^* satisfies the primal complementary slackness condition when it is added to F; this fact is useful for the analysis. Let $C(i)$ and $C(j)$ denote the node sets of the components containing i and j respectively, before the edge $e^* = ij$ is added to F. After ij is added to F, both $C(i)$ and $C(j)$ will satisfy the constraint $|\delta(S) \cap F| \geq f(S)$, since $|\delta(C(i)) \cap F| = 1 \geq f(C(i))$ (similarly for $C(j)$). However, the new connected component that contains the edge $e^* = ij$ and has node set $C = C(i) \cup C(j)$ may have $f(C) = 1$ and so C may now be an active component. The algorithm repeats the above iteration, augmenting edges to F, until F satisfies the proper function f. A sketch of the algorithm follows.

Initialize:
$y_S := 0, \quad \forall S \subset V$; ($y$ satisfies all constraints of the dual LP)
$F := \emptyset$;
while F does not satisfy the given proper function f **do**
let ϵ be the maximum number such that the dual variables y_C of all active components C can be increased to $y_C + \epsilon$ without violating any constraint of the dual LP;
for each active component C, increase y_C by ϵ;
at least one edge $e^* = ij \in \delta(C)$, C an active component, becomes tight, i.e.,
$\sum_{S : e^* \in \delta(S)} y_S$ becomes equal to c_{e^*};
add e^* to F;
end; (while)

perform the clean-up step, and output the resulting forest F';

The key result on the GW algorithm is the following performance guarantee.

Theorem 8.2 *Let $F' \subseteq E$ be the forest output by the GW algorithm, let $y_S(S \subset V)$ be the dual solution at termination of the algorithm, and let Z^*_{IP} be the cost of an optimal solution. Then*

$$\sum_{e \in F'} c_e \le 2 \sum_{S \subset V} y_S \le 2Z^*_{IP}.$$

In other words, the GW algorithm finds a feasible solution whose cost is at most two times the optimal cost. The second inequality in the theorem follows because the objective value of every feasible solution of the dual LP is at most the optimal value of the LP, Z^*_{LP}, by linear programming weak duality, and Z^*_{LP} is at most Z^*_{IP}, since (LP) is a relaxation of (IP).

Before presenting the proof of the first inequality in the theorem, a detailed implementation of the GW algorithm is given, followed by two examples of the working of the algorithm, see Figures 8.1 and 8.2. To obtain a reasonably efficient implementation of the algorithm, two issues need to be examined, namely, how to maintain the dual variables $y_S, S \subset V$, and how to discover the active components in each iteration. The algorithm maintains a list (or set) Γ of the node sets C of the connected components of the current subgraph (V, F). Initially, Γ consists of $|V|$ singleton sets $\{v\}$, one for each $v \in V$. In each iteration, when edge ij is added to F, the components containing the end nodes i and j, namely, $C(i)$ and $C(j)$, "merge", that is, $C(i)$ and $C(j)$ are removed from Γ and are replaced by $C(i) \cup C(j)$. The active components are precisely the node sets C in Γ such that $f(C) = 1$, and these are easily discovered by the algorithm. Let Γ_1 be the set of active components, $\Gamma_1 = \{C \subset V : f(C) = 1 \text{ and } C \text{ is in } \Gamma\}$. Since the algorithm raises the dual variables y_C for the node sets $C \in \Gamma_1$ by ϵ, the dual objective value, $\sum_{S \subset V} f(S) y_S$, increases by $\epsilon \cdot |\Gamma_1|$. In the detailed implementation, there is a variable LB which gives the current dual objective value. Rather than explicitly maintaining all the dual variables, $y_S, S \subset V$, the algorithm maintains a value $d(v)$ for each node v such that

$$d(v) = \sum_{S: v \in S} y_S, \qquad \forall v \in V.$$

Initially, $d(v) = 0$ for each node v, and in each iteration, if the node v is in an active component, then $d(v)$ is increased by ϵ, otherwise $d(v)$ stays unchanged. Clearly, this ensures that the above assertion on $d : V \to \Re_+$ holds at the start of each iteration.

Algorithm GW Constrained Forest Algorithm.
input: Graph $G = (V, E)$, edge costs $c : E \to \Re_+$, proper function $f : 2^V \to \{0, 1\}$.
output: Forest $F' \subseteq E$ satisfying f, and value LB such that LB $\le Z^*_{IP} \le c(F') \le 2 \cdot$ LB ,
where Z^*_{IP} is the cost of an optimal solution.
$F := \emptyset$;
LB $:= 0$;
$\Gamma := \{\{v\} : v \in V\}$;
for each $v \in V$ **do** $d(v) := 0$; **end**;

while $\exists C \in \Gamma$ such that $f(C) = 1$ **do**

(there exists an active component C)
find an edge $e^* = ij$ with $i \in C(i) \in \Gamma$ and $j \in C(j) \in \Gamma$, $C(i) \neq C(j)$,
that minimizes $\epsilon = \dfrac{c_{ij} - d(i) - d(j)}{f(C(i)) + f(C(j))}$;
for each $v \in C \in \Gamma$ **do if** $f(C) = 1$ **then** $d(v) := d(v) + \epsilon$; **end**; **end**;
$\gamma_1 := \sum_{C \in \Gamma} f(C)$; ($\gamma_1$ is the number of active components)
LB := LB $+ \epsilon \cdot \gamma_1$;
$F := F \cup \{ij\}$;
$\Gamma := (\Gamma \backslash \{C(i), C(j)\}) \cup \{(C(i) \cup C(j))\}$;
end (while);

CLEAN-UP:
$F' := \{e \in F : \exists D \subset V : D$ is a connected component of $(V, F\backslash\{e\})$ and $f(D) = 1\}$;

8.6 A performance guarantee for the GW algorithm

The proof of the approximation guarantee for the GW algorithm, Theorem 8.2, is presented here. Recall the statement of the theorem:
Let $F' \subseteq E$ be the forest output by the GW algorithm, let $y_S (S \subset V)$ be the dual solution at termination of the algorithm, and let Z^*_{IP} be the cost of an optimal solution. Then

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subset V} y_S \leq 2Z^*_{IP}.$$

The proof hinges on the next lemma. The lemma characterizes the number of edges of the final output F' (after the CLEAN-UP step) in the "boundary" $\delta(C)$ of a node set C such that $f(C) = 0$. The lemma fails if F' is replaced by F^*, where F^* denotes the forest F just before the CLEAN-UP step.

In the proofs below, a connected component of a subgraph of G (such as the subgraph (V, F)) is sometimes identified with its node set; this should not cause any confusion.

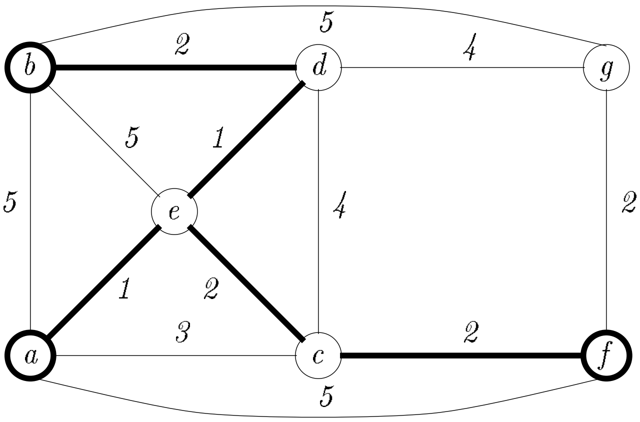
Lemma 8.3 *For every node set $C \subset V$, if $f(C) = 0$, then the final output F' has either zero or two or more edges with exactly one end in C, i.e.,*

$$f(C) = 0 \quad \Longrightarrow \quad |\delta(C) \cap F'| = 0 \quad \text{or} \quad |\delta(C) \cap F'| \geq 2.$$

Proof: The following fact is needed to prove the lemma.

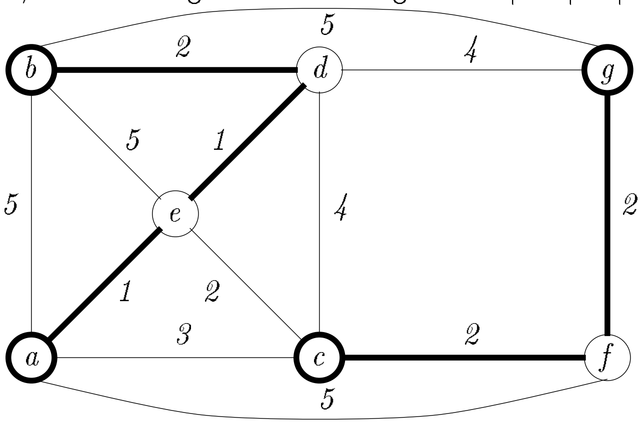
Fact: If S is a subset of V such that $f(S) = 0$, and B is a subset of S such that $f(B) = 0$, then $f(S \backslash B) = 0$.

Figure 8.1: Steiner tree problem on the graph $G = (V, E)$ with terminal node set $N = \{a, b, f\}$ is approximately solved using the GW algorithm. In the list Γ, the active components are indicated by boxes. The final solution F' is indicated in thick lines.



edge $e^* = ij$ added to F	$\epsilon = \frac{c_{ij} - d(i) - d(j)}{f(C(i)) + f(C(j))}$	$d(v)$ $a\ b\ c\ d\ e\ f\ g$	LB	Γ
START $F = \emptyset$	—	0 0 0 0 0 0 0	0	$\boxed{\{\text{a}\}}, \boxed{\{\text{b}\}}, \{c\}, \{d\}, \{e\}, \boxed{\{\text{f}\}}, \{g\}$
ae	$\frac{1-0-0}{1+0} = 1$	1 1 0 0 0 1 0	3	$\boxed{\{\text{a,e}\}}, \boxed{\{\text{b}\}}, \{c\}, \{d\}, \boxed{\{\text{f}\}}, \{g\}$
de	$\frac{1-0-0}{0+1} = 1$	2 2 0 0 1 2 0	6	$\boxed{\{\text{a,e,d}\}}, \boxed{\{\text{b}\}}, \{c\}, \boxed{\{\text{f}\}}, \{g\}$
fg	$\frac{2-2-0}{1+0} = 0$	2 2 0 0 1 2 0	6	$\boxed{\{\text{a,e,d}\}}, \boxed{\{\text{b}\}}, \{c\}, \boxed{\{\text{f, g}\}}$
cf	$\frac{2-0-2}{0+1} = 0$	2 2 0 0 1 2 0	6	$\boxed{\{\text{a,e,d}\}}, \boxed{\{\text{b}\}}, \boxed{\{\text{c, f, g}\}}$
bd	$\frac{2-2-0}{1+1} = 0$	2 2 0 0 1 2 0	6	$\boxed{\{\text{a, b, d, e}\}}, \boxed{\{\text{c, f, g}\}}$
ce	$\frac{2-0-1}{1+1} = \frac{1}{2}$	$\frac{5}{2}\ \frac{5}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{3}{2}\ \frac{5}{2}\ \frac{1}{2}$	7	$\{a, b, c, d, e, f, g\}$
$F^* = \{ae, bd, ce, de, cf, fg\}$				
the CLEAN-UP step removes edge fg final forest: $F' = \{ae, bd, ce, de, cf\}$, $c(F') = 8$.				

Figure 8.2: Generalized Steiner forest problem on the graph $G = (V, E)$ with terminal node sets $N_1 = \{a, b\}$ and $N_2 = \{c, g\}$ is approximately solved using the GW algorithm. The proper function $f : 2^V \to \{0, 1\}$ is given by $f(S) = 1$ if either $\emptyset \neq S \cap N_1 \neq N_1$ or $\emptyset \neq S \cap N_2 \neq N_2$, and $f(S) = 0$ otherwise. In the list Γ, the active components are indicated by boxes. The final solution F' is indicated in thick lines.
F' is an optimal solution, since the edge costs are integral and $\lceil \text{LB} \rceil = \lceil 7.5 \rceil \geq c(F')$.



edge $e^* = ij$ added to F	$\epsilon = \frac{c_{ij} - d(i) - d(j)}{f(C(i)) + f(C(j))}$	$d(v)$ $a\ b\ c\ d\ e\ f\ g$	LB	Γ
START $F = \emptyset$	—	0 0 0 0 0 0 0	0	$\boxed{\{a\}}, \boxed{\{b\}}, \boxed{\{c\}}, \{d\}, \{e\}, \{f\}, \boxed{\{g\}}$
ae	$\frac{1-0-0}{1+0} = 1$	1 1 1 0 0 0 1	4	$\boxed{\{a,e\}}, \boxed{\{b\}}, \boxed{\{c\}}, \{d\}, \{f\}, \boxed{\{g\}}$
ce	$\frac{2-1-0}{1+1} = \frac{1}{2}$	$\frac{3}{2}\ \frac{3}{2}\ \frac{3}{2}\ 0\ \frac{1}{2}\ 0\ \frac{3}{2}$	6	$\boxed{\{a,c,e\}}, \boxed{\{b\}}, \{d\}, \{f\}, \boxed{\{g\}}$
cf	$\frac{2-(3/2)-0}{1+0} = \frac{1}{2}$	2 2 2 0 1 0 2	$\frac{15}{2}$	$\boxed{\{a,c,e,f\}}, \boxed{\{b\}}, \{d\}, \boxed{\{g\}}$
fg	$\frac{2-0-2}{1+1} = 0$	2 2 2 0 1 0 2	$\frac{15}{2}$	$\boxed{\{a,c,e,f,g\}}, \boxed{\{b\}}, \{d\}$
de	$\frac{1-0-1}{0+1} = 0$	2 2 2 0 1 0 2	$\frac{15}{2}$	$\boxed{\{a,c,d,e,f,g\}}, \boxed{\{b\}}$
bd	$\frac{2-2-0}{1+1} = 0$	2 2 2 0 1 0 2	$\frac{15}{2}$	$\{a, b, c, d, e, f, g\}$
$F^* = \{ae, bd, ce, de, cf, fg\}$				
the CLEAN-UP step removes edge ce final forest: $F' = \{ae, bd, de, cf, fg\}$, $c(F') = 8$.				

Proof: (of the fact) Consider the set $S \backslash B$, and note that its complement, $V \backslash (S \backslash B)$ is the union of two disjoint sets $V \backslash S$ and B. Now,

$$\begin{aligned}
f(S \backslash B) &= f(V \backslash (S \backslash B)), \quad \text{by symmetry,} \\
&= f((V \backslash S) \cup (B)) \\
&\leq f(V \backslash S) + f(B), \quad \text{by maximality,} \\
&= f(S) + f(B), \quad \text{since } f(V \backslash S) = f(S) \text{ by symmetry,} \\
&= 0 + 0.
\end{aligned}$$

▮

The lemma is proved by contradiction. Suppose that there is a node set C such that $f(C) = 0$ and $|\delta(C) \cap F'| = 1$. Let e^* be the unique edge of F' with exactly one end in C. Let F^* denote the forest F just before the CLEAN-UP step. Let C^* be the node set of the connected component of the forest F^* that contains C, and let the node sets of the two connected components formed from C^* by removing e^* be N^* and $(C^* \backslash N^*)$. Fix the notation such that C is a subset of N^*. There are two cases.

case (1): $N^* = C$.
Then $f(N^*) = f(C) = 0$. Further, $f(C^*) = 0$, since by the definition of F^*, the node set D of each connected component of (V, F^*) has $f(D) = 0$. From the above fact it follows that $f(C^* \backslash N^*) = 0$. This gives the desired contradiction, since the CLEAN-UP step should have removed the edge e^* from F'.

case (2): $N^* \supset C$.
Then $\delta(C) \cap F^*$ has one or more edges, besides the edge e^*. Let $e^*, e_1, \ldots, e_k$ be the edges in $\delta(C) \cap F^*$. Removing each of the edges e_i, $1 \leq i \leq k$, from F^* disconnects C^* into two connected components, one containing C and one contained in N^*. Let $C_1, \ldots, C_k$, $C_i \subset N^* (1 \leq i \leq k)$, denote the node sets of the latter kind of connected components of $(V, F^* \backslash e_i)$, $1 \leq i \leq k$. Clearly, the node sets $C, C_1, \ldots, C_k$ are (pairwise) disjoint, and N^* is the union of these node sets. Since the CLEAN-UP step removes the edges $e_1, \ldots, e_k$ from F^* to obtain F', it follows that $f(C_i) = 0$ for each i, $1 \leq i \leq k$. Hence, by maximality,

$$f(N^*) \leq f(C) + f(C_1) + \ldots + f(C_k) = 0.$$

Now, the proof is completed as in case (1).

▮

Proof: (**Theorem 8.2**) For every edge e added by the algorithm to F

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

That is, each edge e added to F satisfies the primal complementary slackness condition. Hence,

$$\begin{aligned}
\sum_{e \in F'} c_e &= \sum_{e \in F'} \sum_{S: e \in \delta(S)} y_s \\
&= \sum_{S \subset V} y_S \cdot |\delta(S) \cap F'|.
\end{aligned}$$

The proof is completed by using induction on the number of iterations to show that

$$\sum_{S \subset V} y_S \cdot |\delta(S) \cap F'| \quad \leq \quad 2 \sum_{S \subset V} y_S. \tag{1}$$

The induction basis holds, since at the start of the algorithm $y_S = 0, \ \forall S \subset V$, hence inequality (1) holds. For the induction step, note that in each iteration, for each active component C in the list Γ, y_C increases by ϵ, and for all remaining sets $S \subset V$, y_S stays the same. That is, the left-hand side of inequality (1) increases by

$$\epsilon \sum_{C:C \in \Gamma} f(C) \cdot |\delta(C) \cap F'|,$$

and the right-hand side of inequality (1) increases by

$$2\epsilon \sum_{C:C \in \Gamma} f(C).$$

Inequality (1) would continue to hold if

$$\sum_{C:C \in \Gamma} f(C) \cdot |\delta(C) \cap F'| \quad \leq \quad 2 \sum_{C:C \in \Gamma} f(C), \tag{2}$$

that is, if the average number of edges of F' in the "boundary" $\delta(C)$ of an active component C is at most two.

The proof is completed by showing that inequality (2) holds. Construct a graph H from the list Γ: the nodes of H correspond to the node sets in Γ, and there is an edge CD in H iff in the original graph $G = (V, E)$ there is an edge $e \in F'$ such that $e \in \delta(C)$ and $e \in \delta(D)$. That is, the graph H is obtained from the subgraph (V, F') of G by contracting each node set C in the list Γ.
Fact: The graph H is a forest.

For the rest of the proof, assume that no tree of H has a single node, i.e., assume that in the original graph G each node set C in the list Γ has $|\delta(C) \cap F'| \geq 1$. The proof is easily modified for the case when the assumption does not hold.

Since H is a forest, it has at most $|V(H)| - 1$ edges, and so the sum of the degrees of the nodes in H is at most $2(|V(H)| - 1)$. In terms of the original graph G this gives

$$\sum_{C \in \Gamma} |\delta(C) \cap F'| \quad \leq \quad 2(|\Gamma| - 1) \tag{3}$$

From Lemma 8.3 and the assumption above, it follows that in the original graph G for each node set C in the list Γ with $f(C) = 0$, $|\delta(C) \cap F'|$ is at least two. Summing over all node sets C in Γ with $f(C) = 0$ gives

$$\sum_{C \in \Gamma} (1 - f(C)) \cdot |\delta(C) \cap F'| \quad \geq \quad 2 \sum_{C \in \Gamma} (1 - f(C)),$$

or, multiplying through by -1,

$$\sum_{C \in \Gamma} (f(C) - 1) \cdot |\delta(C) \cap F'| \quad \leq \quad 2 \sum_{C \in \Gamma} (f(C) - 1). \tag{4}$$

Adding inequalities (3) and (4) gives the desired inequality (2).

The proof of the induction step is now complete, and so the theorem has been proved. ∎

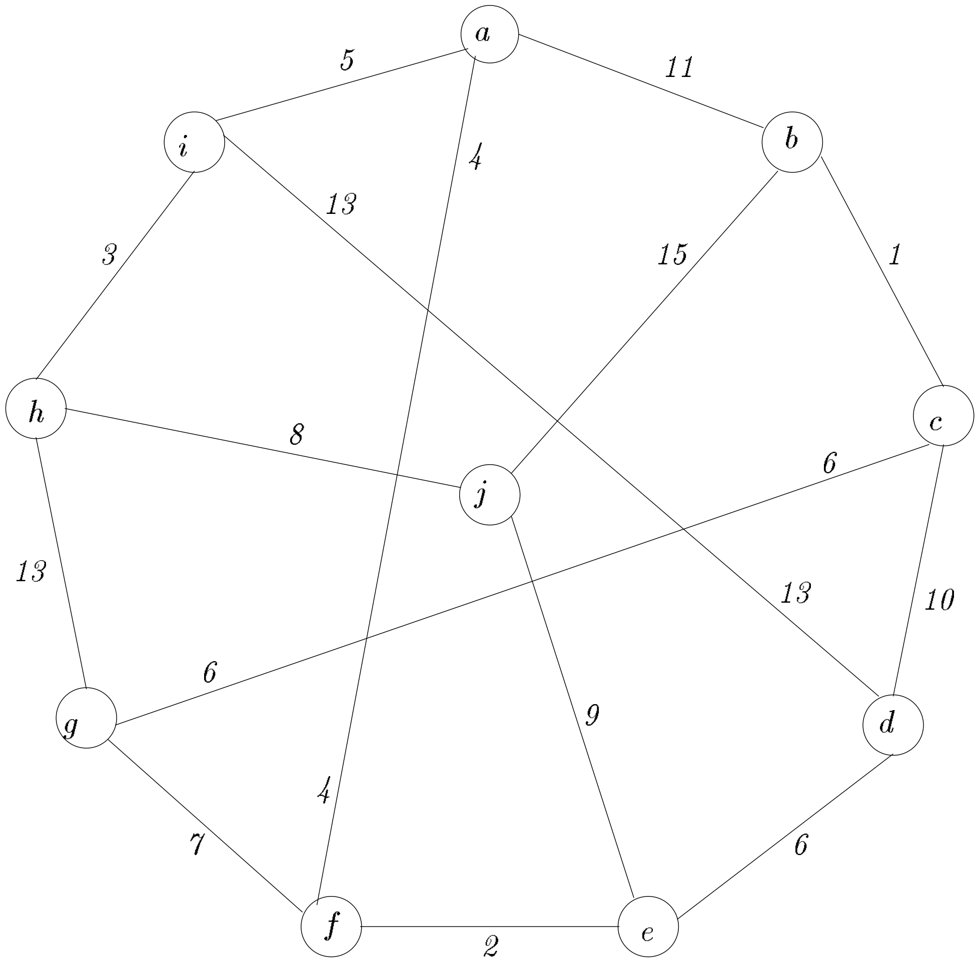


Figure 8.3: A network $G,\ c$ for constrained forest problems.

8.7 Exercises

1. Let f be a proper function, and let $S^* \subset V$ be a set maximizing $f(S^*)$. Prove that there is a node $v \in S^*$ such that $f(\{v\}) = f(S^*)$.

2. A function $f : 2^V \to \{0,1\}$ is said to satisfy the *complementarity* property if for all $A \subseteq S \subseteq V$, $f(S) = f(A) = 0 \implies f(S \backslash A) = 0$.

   (a) Prove that for symmetric functions f, the maximality property is equivalent to the complementarity property.

   (b) Let $f : 2^V \to \{0,1\}$ be a (not necessarily symmetric) function satisfying maximality and complementarity. Show that the symmetrization f_{sym} is a proper function, where f_{sym} is given by
$$f_{sym}(S) = \max(f(S),\ f(V \backslash S)), \qquad \forall S \subseteq V.$$

3. A function $h : 2^V \to \{0,1\}$ is said to be *uncrossable* if $h(V) = 0$, and
$$\forall A \subseteq V, B \subseteq V \qquad h(A) = h(B) = 1 \quad \implies \quad \begin{array}{l} \text{either } h(A \cup B) = h(A \cap B) = 1, \text{ or} \\ h(A \backslash B) = h(B \backslash A) = 1. \end{array}$$
   Prove that any function $g : 2^V \to \{0,1\}$ satisfying maximality is uncrossable.

4. In each of parts (ii), (iii) and (iv) of Section 8.3, prove that the function f is proper.

5. Let $G = (V, E)$ be a graph, and let $c : E \to \Re_+$ be a nonnegative cost function on the edges. The goal is to use the GW (Goemans-Williamson) algorithm to find a minimum spanning tree (MST) of $G,\ c$.

   (a) State the function $f : 2^V \to \{0,1\}$ to use for finding an MST, and prove that the function f is proper.

   (b) Explain why the following is true:
   If the *GW* algorithm is applied to $G,\ c$ with an appropriate proper function f, then it finds a minimum spanning tree.

6. Use the constrained forest algorithm of Goemans and Williamson to solve the following problems on the network $G,\ c$ in Figure 8.3.

   (a) Find (an approximate) Steiner minimal tree with terminal node set $N = \{b, e, h\}$.

   (b) Solve (approximately) the generalized Steiner forest problem with $N_1 = \{a, b, e\}$ and $N_2 = \{d, j\}$.

   (c) Solve (approximately) the point-to-point connection problem with the source set $C = \{b, e, f\}$ and the destination set $D = \{a, h, j\}$.

   (d) Solve (approximately) the T-join problem with $T = \{a, b, e, f, h, j\}$.

Bibliography

[1] A. Agrawal, P. N. Klein and R. Ravi, *When trees collide: An approximation algorithm for the generalized Steiner problem on networks*, SIAM J. Computing **24** (1995). Preliminary version in Proc. 23rd Ann. ACM S. T. O. C. (1991), pp. 134–144.

[2] F. A. Chudak, M. X. Goemans, D. S. Hochbaum, and D. P. Williamson, *A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs.* To appear in Operations Research Letters. Available at `http://www.research.ibm.com/people/w/williamson/`

[3] M. X. Goemans and D. P. Williamson, *A general approximation technique for constrained forest problems*, SIAM J. Computing **24** (1995).

[4] M. X. Goemans and D. P. Williamson, *The primal-dual method for approximation algorithms and its application to network design problems*, in *Approximation algorithms for NP-hard problems*, ed. D. S. Hochbaum, PWS publishing co., Boston, 1996.

[5] M. Grötschel, C. L. Monma and M. Stoer, *Design of survivable networks*, chapter from the Handbook in Operations Research and Management Science, March 1993.

[6] F. K. Hwang, D. S. Richards and P. Winter, *The Steiner Tree Problem.* Annals of Discrete Mathematics, 53, North-Holland, Amsterdam, 1992.

[7] K. Jain, *A factor 2 approximation algorithm for the generalized Steiner network problem.* Manuscript, January, 1998, `http://www.cc.gatech.edu/people/home/kjain/`
To appear in the Proc. 39th IEEE FOCS, Palo Alto, CA, November 1998.

[8] T. L. Magnanti and L. A. Wolsey, *Optimal Trees.* CORE discussion paper 9426, C. O. R. E., Universite Catholique de Louvain, Louvain-la-neuve, Belgium, May 1994.

[9] R. Ravi and D. P. Williamson, *An approximation algorithm for minimum-cost vertex-connectivity problems. Algorithmica* **18** (1997), 21-43.

[10] M. Stoer and G. Dahl, *A polyhedral approach for multicommodity survivable network design*, Numerische Mathematik **68** (1994), pp. 149–167.

[11] D. P. Williamson, M. X. Goemans, M. Mihail and V. V. Vazirani, *A primal-dual approximation algorithm for generalized Steiner network problems*, Proceedings 25th ACM Symposium on Theory of Computing, 1993.