CS 684: Approximation and Network Algorithms	**October 19, 2001**
Primal-Dual Approximation for Network Design Problems	**David Kempe**

1 The Network Design Problem

In the general network design problem, we are given an (undirected) graph $G = (V, E)$ with edge costs $c_e \geq 0$, and connectivity requirements $r_{ij} \geq 0$ for each pair $i, j \in V$ of vertices.[1] We are looking for a subset $F \subseteq E$ of edges of minimum total cost such that in the graph (V, F), all pairs i, j of nodes are at least r_{ij}-connected.

Note that there are several equivalent characterizations of connectivity — in particular, the following are equivalent:

1. i and j are r-connected.
2. All i-j-cuts have at least r edges.
3. i and j are connected in $(V, F \setminus F')$ for any edge set F' with $|F'| < r$.
4. There are r edge-disjoint paths between i and j.

The equivalence of the first and second characterization is obvious, whereas the third one follows from the Max-Flow Min-Cut Theorem.

For the remainder, we will restrict our attention to the case that all r_{ij} are either 0 or 1. Even in this restricted version, the problem is NP-complete, since it contains the Graph Steiner Tree problem. In the Graph Steiner Tree Problem, we are given a set of terminal nodes $X \subseteq V$, and want to find a minimum cost subtree T of G such that all terminals $v \in X$ are connected in T. For that purpose, we may include additional vertices $w \in V \setminus X$ into the tree T. This problem is well known as an NP-complete problem, and is a special case of the general network design problem by setting $r_{ij} = 1$ for all $i, j \in X$, and $r_{ij} = 0$ whenever $i \notin X$ or $j \notin X$.

Since the network design problem is NP-complete, we will not be able to solve it optimally, and instead will present a primal-dual 2-approximation algorithm.

2 IP, LP, and pricing

A straightforward integer programming formulation of the network design problem assigns (decision) variables $x_e \in \{0, 1\}$ to each edge e, and asks us to

$$\begin{array}{lllll} \text{Minimize} & \sum_e x_e c_e & & & \\ \text{subject to} & \sum_{e \in \delta(S)} x_e & \geq & f(S) := \max_{i \in S, j \notin S} r_{ij} & \text{for all } S \subseteq V \end{array}$$

Here, $\delta(S)$ denotes the set of edges with exactly one endpoint in S. We obtain the LP-relaxation in the obvious way by changing the requirement $x_e \in \{0, 1\}$ to $x_e \in [0, 1]$.

To obtain a primal-dual algorithm, we want to develop an "economical" pricing scheme assigning prices p_S to sets S with $f(S) > 0$. This pricing scheme will be a meaningful interpretation of the dual program to the above LP. In light of our previous primal-dual schemes, the pricing scheme should satisfy the following conditions (which can be derived mechanically from the Duality and the Complementary Slackness Theorem):

(1) $\sum_{S: e \in \delta(S)} p_S \leq c_e$ for all edges e. No edge is overpaid by the pricing scheme.

[1] We assume that the requirements are symmetric, i.e. $r_{ij} = r_{ji}$ for all i, j.

(2) If $e \in F$, then $\sum_{S:e\in\delta(S)} p_S = c_e$. If an edge is included in the solution, then it is fully paid for.

(3) If $p_S > 0$, then $|F \cap \delta(S)| = 1$. If a set pays any amount, it has only one edge leaving it, so we know "who to make the payment to". This of course applies only in our special case that $r_{ij} \in \{0,1\}$ for all i, j.

If F and p satisfy all of the above conditions, we say that they are *at equilibrium*. If we had p and F which are at equilibrium, then p would be proof of F's optimality, and since the problem is NP-complete, we cannot expect to compute such a scheme p in polynomial time. Hence, we want to relax one of the constraints to make the problem more tractable. We will retain the first two constraints, and change the third one to

(3′) If $p_S > 0$, then $|F \cap \delta(S)| \leq 2$.

If we had p and F satisfying conditions (1), (2) and (3′), we can see that F is a 2-approximation to the optimum network.

Lemma 2.0.1 *If F and p satisfy constraints (1), (2) and (3′), and F^* is an optimal solution to the network design problem, then $c_F \leq 2 \cdot c_{F^*}$.*

Proof. An optimal solution F^* satisfies

$$\begin{aligned}
\sum_{e\in F^*} c_e &\overset{(1)}{\geq} \sum_{e\in F^*} \sum_{S:e\in\delta(S)} p_S \\
&= \sum_S p_S \cdot |F^* \cap \delta(S)| \\
&\geq \sum_S p_S.
\end{aligned}$$

The last inequality holds because $|F^* \cap \delta(S)| \neq 0$ for all sets S with $p_S > 0$. On the other hand, F satisfies (2) and (3′), so

$$\begin{aligned}
\sum_{e\in F} c_e &\overset{(2)}{=} \sum_{e\in F} \sum_{S:e\in\delta(S)} p_S \\
&= \sum_S p_S \cdot |F \cap \delta(S)| \\
&\overset{(3')}{\leq} \sum_S 2 \cdot p_S,
\end{aligned}$$

which we know to be at most $2 \cdot c_{F^*}$, completing the proof. ■

3 A primal-dual 2-approximation algorithm

The following primal-dual algorithm is due to Goemans and Williamson. It actually does not quite maintain condition (3′) from above, only an "averaged" version that will suffice to prove the desired 2-approximation guarantee. The algorithm consists of two phases, a growing phase and a cleanup phase.

In the growing phase, we maintain a set F of edges, and in each iteration increase the prices p_S for "unsatisfied" (or *active*) sets, i.e. sets which have at least one connectivity requirement that has not yet been met. Once equality holds for some edge e in condition (1), i.e. $\sum_{S:e\in\delta(S)} p_S = c_e$, we call e *tight* and add it to F.

Let $F^{(k)}$ denote the set of edges that are selected at the end of iteration k, and $\mathcal{C}^{(k+1)}$ the set of connected components induced by $F^{(k)}$ on V. Let $\mathcal{M}^{(k+1)}$ be those components in $\mathcal{C}^{(k+1)}$ which have at least one unsatisfied requirement. That is, $S \in \mathcal{M}^{(k+1)}$ if there is some $i \in S$ and $j \notin S$ with $r_{ij} = 1$. We call the components $S \in \mathcal{M}^{(k+1)}$ *active* components. The prices assigned to components S after iteration k are denoted by $p_S^{(k)}$. The growing phase is shown in Figure 1. Notice that the algorithm maintains conditions (1) and (2) explicitly.

```
1.  Start with p_S^(0) := 0 for all S, F^(0) := ∅.
2.  Let k := 1.
3.  While M^(k) ≠ ∅ (there are active components) do
    4.  Let Δ^(k) be the minimum Δ such that
          increasing p_S^(k) by Δ for all active sets S ∈ M^(k)
          will make at least one additional edge tight.
    5.  Let p_S^(k) := p_S^(k-1) + Δ^(k) for all active sets S ∈ M^(k).
    6.  Let F^(k) := {e ∈ E | Σ_{S:e∈δ(S)} p_S^(k) = c_e}.
```

Figure 1: The growing phase of the primal-dual algorithm

In the special case that $r_{st} = 1$ for only two specific nodes $s, t \in V$, and $r_{ij} = 0$ for $\{i, j\} \neq \{s, t\}$, the network design problem becomes equivalent to the shortest path problem. In that case, the algorithm will increase prices only for components containing either s or t, and hence becomes Dijkstra's shortest path algorithm, run from the source and sink simultaneously. Notice, however, that the output of the growing phase is a shortest-paths tree which may contain many unnecessary edges. For this purpose, the algorithm has a second cleanup phase, which is described in Figure 2.

```
1.  Start with the final output F of the growing phase.
2.  Consider all edges e ∈ F in the reverse insertion order,
     and remove e from F if F \ {e} still satisfies all connectivity requirements.
```

Figure 2: The cleanup phase of the primal-dual algorithm

Figure 3 illustrates the growing phase on a six-node graph. The costs on the edges are just their lengths in the two-dimensional drawing. The connectivity requirements are given by the following matrix

$$R = (r_{ij}) = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

In the drawings, medium dark edges are part of the set $F^{(k)}$ after iteration k, and very dark edges are the ones that have become tight in iteration k. The shaded areas around nodes (or node sets) illustrate the prices $p_S^{(k)}$ for those sets. The nodes in active components have the node numbers written in them; those nodes that have become inactive during the iteration are depicted by empty circles.

The key ingredient to proving the primal-dual algorithm to be a 2-approximation is a lemma that shows that it maintains an "averaged" version of property (3′).

Lemma 3.0.2 *Let F be the final output (after the cleanup phase), and $\mathcal{M}^{(k)}$ the set of active components in some iteration k. Then, $\sum_{S \in \mathcal{M}^{(k)}} |\delta(S) \cap F| \leq 2 \cdot |\mathcal{M}^{(k)}|$.*

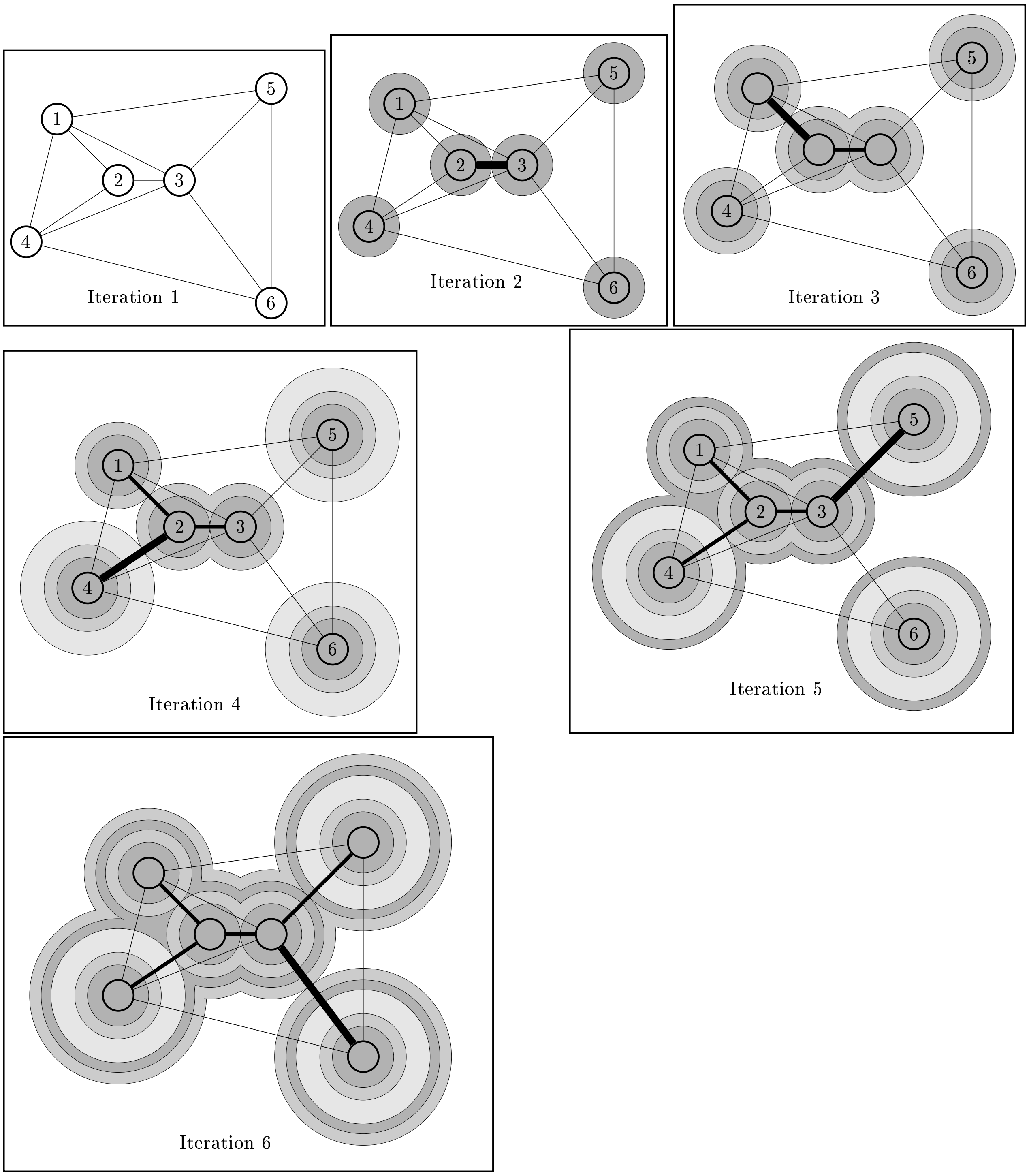

Figure 3: The growing phase in 6 iterations.

Proof. First, we want to show that if we contract all $S \in \mathcal{C}^{(k)}$ to single nodes, the graph induced by F on these nodes is acyclic. By definition, all components $S \in \mathcal{C}^{(k)}$ were connected before any edges between different components are added to F. Hence, if any cycles had been formed at later stages, the cleanup phase would have deleted those edges that formed a cycle (since the connectivity requirements are satisfied without them). So the contracted graph is actually a forest.

Next, we claim that in this forest, all inactive components $S \notin \mathcal{M}^{(k)}$ have at least 2 crossing edges, i.e. $|\delta(S) \cap F| \geq 2$. For if an inactive component had degree 1, the edge incident with it could have been deleted in the cleanup phase without violating the connectivity requirements. Therefore, all leaves in the forest resulting from the contraction are active.

Because the sum over all degrees in a forest is at most twice the number of nodes, we obtain that $\sum_{S \in \mathcal{C}^{(k)}} |\delta(S) \cap F| \leq 2 \cdot |\mathcal{C}^{(k)}|$. Hence,

$$\begin{aligned}\sum_{S \in \mathcal{M}^{(k)}} |\delta(S) \cap F| &\leq 2 \cdot |\mathcal{C}^{(k)}| - \sum_{S \in \mathcal{C}^{(k)} \setminus \mathcal{M}^{(k)}} |\delta(S) \cap F| \\ &\leq 2 \cdot |\mathcal{C}^{(k)}| - 2 \cdot |\mathcal{C}^{(k)} \setminus \mathcal{M}^{(k)}| \\ &= 2 \cdot |\mathcal{M}^{(k)}|,\end{aligned}$$

completing the proof. ■

We can now state and prove the main Theorem guaranteeing the approximation ratio of 2.

Theorem 3.0.3 *The above primal-dual algorithm is a 2-approximation for the network design problem with requirements $r_{ij} \in \{0, 1\}$.*

Proof. The lower bound on the cost of an optimum solution F^* is unchanged from the proof of Lemma 2.0.1, and hence, we know that $\sum_{e \in F^*} c_e \geq \sum_S p_S$.

For an upper bound on the cost of the solution F returned by the algorithm, we divide the costs into the contributions made by the different phases of the growing process. More specifically, for any set S, we know that $p_S = \sum_{k : S \in \mathcal{M}^{(k)}} \Delta^{(k)}$. We thus obtain that

$$\begin{aligned}\sum_{e \in F} c_e &\stackrel{(2)}{=} \sum_{e \in F} \sum_{S : e \in \delta(S)} p_S \\ &= \sum_S p_S \cdot |F \cap \delta(S)| \\ &= \sum_k \Delta^{(k)} \cdot \sum_{S \in \mathcal{M}^{(k)}} |F \cap \delta(S)| \\ &\stackrel{\text{Lemma 3.0.2}}{\leq} \sum_k \Delta^{(k)} \cdot 2 \cdot |\mathcal{M}^{(k)}| \\ &= 2 \cdot \sum_S p_S \\ &\leq 2 \cdot \mathrm{OPT},\end{aligned}$$

completing the proof of the Theorem. ■