Greedy Approximation Algorithms for Directed Multicuts

Yana Kortsarts
Department of Computer Science, Widener University, Chester, Pennsylvania 19013

Guy Kortsarz
Department of Computer Science, Rutgers University, Camden, New Jersey 08102

Zeev Nutov
Open University of Israel, Klauzner 16 Str., Ramat-Aviv, Israel

The Directed Multicut (DM) problem is: given a simple directed graph $G = (V, E)$ with positive capacities u_e on the edges, and a set $K \subseteq V \times V$ of ordered pairs of nodes of G, find a minimum capacity K-multicut; $C \subseteq E$ is a K-multicut if in $G - C$ there is no (s, t)-path for any $(s, t) \in K$. In the uncapacitated case (UDM) the goal is to find a minimum size K-multicut. The best approximation ratio known for DM is $O(\min\{\sqrt{n}, opt\})$ by Gupta, where $n = |V|$, and opt is the optimal solution value. All known nontrivial approximation algorithms for the problem solve large linear programs. We give the first combinatorial approximation algorithms for the problem. Our main result is an $\tilde{O}(n^{2/3}/opt^{1/3})$-approximation algorithm for UDM, which improves the $O(\min\{opt, \sqrt{n}\})$-approximation for $opt = \Omega(n^{1/2+\varepsilon})$. Combined with the article of Gupta, we get that UDM can be approximated within better than $O(\sqrt{n})$, unless $opt = \tilde{\Theta}(\sqrt{n})$. We also give a simple and fast $O(n^{2/3})$-approximation algorithm for DM. © 2005 Wiley Periodicals, Inc. NETWORKS, Vol. 45(4), 214–217 2005

Keywords: directed; graphs; multicuts; approximation

1. INTRODUCTION AND PRELIMINARIES

An instance of the *Directed Multicut* (DM) problem consists of a *simple* directed graph $G = (V, E)$ with integral capacities u_e on the edges and a set $K \subseteq V \times V$ of ordered pairs of nodes of G. The goal is to find a minimum K*-multicut*, that is, a minimum capacity edge set C so that in $G - C$ there is no (s, t)-path for any $(s, t) \in K$. In the *uncapacitated case* (UDM), all edges have capacity 1.

Received June 2004; accepted March 2005
Correspondence to: G. Kortsarz; e-mail: guyk@camden.rutgers.edu
DOI 10.1002/net.20066
Published online in Wiley InterScience (www.interscience.wiley.com).

The minimum multicut problem is one of the most fundamental problems in optimization. Edge (and vertex) cuts are important for the study of Markov chains and geometric embedding. They also appear in the study of clustering, divide and conquer approaches, PRAM emulation, VLSI layout, and packet routing in distributed networks (see, e.g., [4] and references therein). The directed multicut problem is the dual of the fundamental multicommodity flow problem. See Chapter 5 of the book [10] for more details on multicut problems in the context of approximation algorithms. Although the undirected graph case enjoys some efficient approximation algorithms (see references below) the directed case seems much harder. Few approximation results exist for the directed case despite the considerable attention it has received.

We survey some related work. The case $|K| = 1$ is polynomially solvable based on the fundamental Max-Flow Min-Cut Theorem. For $|K| > 1$ the min-cut max-flow equality breaks down even on undirected graphs. In fact, the undirected multicut problem is MAXSNP-hard even on stars [8]. A 2-approximation algorithm is given in [8] for the undirected multicut problem on trees. The best approximation ratio for the minimum multicut problem on general undirected graphs is $O(\log |K|)$ [7].

In [11], a related problem is studied. The input is as in the DM problem, except that the pairs in K are unordered. The goal is to remove a min-capacity edge set C so that in $G - C$ no cycle contains a pair from K. This problem seems easier than the DM problem. In particular, divide-and-conquer methods similar to the ones in [4, 7, 12] give an $O(\log^2 |K|)$-approximation for this variant [11]. In [4], a relatively general scheme is presented handling many problems that are “decomposable,” but DM does not seem to lend itself in any way to the divide-and-conquer approach. Given this fact, it may be that the directed multicut problem is harder to approximate than the undirected one. In particular, a (poly)logarithmic approximation is not known for DM, nor for UDM. However, so far, an exact proof separating the approximability of the undirected and directed problems does not exist. In fact, the only approximation threshold known for the directed case is the one derived from the undirected case: namely, that the problem is MAXSNP-hard.

The first nontrivial approximation ratio of $O(\sqrt{n \log n})$ for DM is due to Cheriyan et al. [1]. This was slightly improved by Gupta [9] to $O(\sqrt{n})$. Gupta's analysis also gives an $O(opt^2)$ capacity solution, with *opt* being the optimal multicut capacity. This can be considered as an $O(opt)$-approximation algorithm, and is useful when *opt* is "small." Both algorithms [1] and [9] require solving large linear programs.

We design *combinatorial* approximation algorithms for DM. Let n and m be the number of nodes and edges, respectively, in the input graph. We use the $\tilde{O}$ notation, which ignores polylogarithmic factors. Our main result is:

Theorem 1.1. *For UDM there exists an algorithm with running time $\tilde{O}(n^2 m)$ that finds a multicut C of size $O((n \log n \cdot opt)^{2/3}) = \tilde{O}((n \cdot opt)^{2/3})$.*

The approximation ratio is $\tilde{O}(n^{2/3}/opt^{1/3})$. Therefore, Theorem 1.1 implies that for UDM the $\sqrt{n}$-approximation can be improved if *opt* is large (e.g., $opt = \Omega(n^{1/2+\varepsilon})$ for some $\varepsilon > 0$). This is the first algorithm whose approximation ratio *improves* as *opt* gets larger. Combined with the results of [9], which provide an $O(opt)$-approximation, we get an approximation ratio better than $\tilde{O}(\sqrt{n})$, unless $opt = \tilde{\Theta}(\sqrt{n})$.

Our additional result is:

Theorem 1.2. *DM admits an $O(n^{2/3})$-approximation algorithm with running time $\tilde{O}(nm^2)$.*

The approximation ratio in [1, 9] is better than the one in Theorem 1.2. However, our algorithm is very simple, and runs faster than the algorithms in [1, 9]; the latter can be implemented in $O(n^2 m^2)$ time using the algorithm of Fleischer [5] for finding an approximate solution of multicommodity-flow type linear programs.

We prove Theorems 1.1 and 1.1 in Sections 2 and 3, respectively.

We now describe the notation used. Let $G = (V, E)$ be a directed graph. For $s, t \in V$ the *distance* $d_G(s, t)$ from s to t in G is the minimum number of edges in an (s, t)-path; $d_G(s, t) = \infty$ if no (s, t)-path exists in G. For disjoint subsets $S, T \subseteq V$ of V let $\delta_G(S, T) = \{st \in E : s \in S, t \in T\}$. We often omit the subscript G if it is clear from the context. An edge set $C \subseteq E$ is an (s, t)-cut if $C = \delta(S)$ for some $S \subseteq V - t$ with $s \in S$. Let $u(C) = \sum\{u_e : e \in C\}$ be the capacity of C; $u(C) = |C|$ if no capacities are given. For simplicity of exposition, we ignore that some numbers are not integral. The adaptation using floors and ceilings is immediate. Unless specifically stated otherwise, all logs in the article are to the base 2.

Before we describe the algorithm, a few preliminary remarks are required. Our algorithms run with certain parameters, which should get appropriate values that depend on n and *opt* to achieve the claimed approximation ratios. Specifically, for UDM we show an algorithm that for any integer ℓ computes a multicut of size $\ell \cdot opt + O((n \log n)^2/\ell^2)$. Setting $\ell = (n \log n)^{2/3}/opt^{1/3}$ gives the claimed approximation ratio. Because *opt* is not known, we execute the algorithm for $\ell = 1, \dots, (n \log n)^{2/3}$, and among the multicuts computed output one of minimum size. For DM, we show an algorithm that for any integers ℓ, μ with $1 \leq \ell \leq n - 1$ and $\mu \geq opt$ computes a K-multicut of capacity $\leq \mu \cdot (2\ell + n^2/\ell^2)$. Setting $\ell = n^{2/3}$ and $\mu = opt$ gives the claimed approximation ratio. Since *opt* is not known, we apply binary search to find the minimum integer μ so that a multicut of capacity $\leq \mu \cdot (2\ell + n^2/\ell^2)$ is returned. Note that if $\mu \geq opt$, a multicut C of capacity $\leq \mu(2\ell + n^2/\ell^2)$ is returned. If $\mu < opt$, then either the returned multicut C is of capacity $\leq \mu(2\ell + n^2/\ell^2) < 3(opt \cdot n)^{2/3}$, which is fine or we know that $\mu < opt$ as the above inequality fails.

Remark. Recently we became aware of the article [13], which gives an $\tilde{O}(n^{2/3})$-approximation algorithm for the related Edge-Disjoint Paths problem. Our result for UDM, which was derived independently, and the main result in [13] rely on the same combinatorial statement (Corollary 2.5 in our article, Theorem 1.1 in [13]), but the proofs are different.

2. THE UNCAPACITATED CASE

Definition 2.1. *For $X, Y \subseteq V$, let $R_G(X, Y) = |\{(x, y) \subseteq X \times Y : x \neq y, d_G(x, y) < \infty\}|$ denote the number of pairs $(x, y) \subseteq X \times Y$ such that an (x, y)-path exists; let $R(G) = R_G(V, V)$.*

Definition 2.2. *We say that $G = (V, E)$ is a p-layered graph if V can be partitioned into p layers $L_1, \dots, L_p$ so that every $e \in E$ belongs to $\delta_G(L_i, L_{i+1})$ or to $\delta_G(L_i, L_i)$ for some $i \in \{1, \dots, p-1\}$, or to $\delta_G(L_j, L_i)$ for some $i \in \{1, \dots, p-1\}$, $j \in \{2, \dots, p\}$, $j > i$.*

Lemma 2.1. *Let $G = (V, E)$ be a 4-layered graph containing at least k edge-disjoint (L_1, L_4)-paths such that $G - \delta_G(L_2, L_3)$ is a simple graph. Then $R(L_1, L_3) + R(L_2, L_4) \geq k$.*

Remark. Observe that the graph induced by $L_2 \cup L_3$ may contain parallel edges.

Proof. We will prove the statement by induction on k. The case $k = 0$ is obvious. Assume $k \geq 1$, and that E is a union of the k edge-disjoint paths. Let $st \in \delta_G(L_2, L_3)$, let $G' = G - \{s, t\}$, and let $S = \{v \in L_1 : vs \in E\}$, $T = \{v \in L_4 : tv \in E\}$. Then G' contains at least $k - (|S| + |T|)$ edge-disjoint (L_1, L_4)-paths. Also, $R_{G'}(L_1, L_3) \leq R_G(L_1, L_3) - |S|$ and $R_{G'}(L_2, L_4) \leq R_G(L_2, L_4) - |T|$. This follows because of the removal of $\{s, t\}$. By the induction hypothesis, if $k > |S| + |T|$, then $R_{G'}(L_1, L_3) + R_{G'}(L_2, L_4) \geq k - (|S| + |T|)$. Thus, $R_{G'}(L_1, L_3) + R_{G'}(L_2, L_4) \geq \max\{k - (|S| + |T|), 0\}$. Combining, we get the statement. ■

Lemma 2.2. *Let G be a simple ℓ-layered graph containing k edge-disjoint paths from the first layer to the last layer, and let*

S and T be the union of the $p_S \geq 2$ first and $p_T \geq 2$ last layers, respectively, so that $S \cap T = \emptyset$. Then $R(S,T) = \Omega(kp_Sp_T)$.

Proof. By Lemma 2.1, $R(L_i, L_j) + R(L_{i+1}, L_{j+1}) \geq k$ for every two pairs $L_i, L_{i+1} \subseteq S$ and $L_j, L_{j+1} \subseteq T$. This is shown as follows. Start with the graph induced by $L_i \cup L_{i+1} \cup L_j \cup L_{j+1}$. Let $\mathcal{P}$ be one of the k paths guaranteed by the premise in the lemma. Associate with $\mathcal{P}$ a pair of nodes u_{i+1}, u_j with $u_{i+1} \in L_{i+1}$ and $u_j \in L_j$. The u_{i+1} vertex is the first L_{i+1} vertex in $\mathcal{P}$ and u_j is the last L_j vertex in $\mathcal{P}$. For every pair $u_{i+1} \in L_{i+1}, u_j \in L_j$ put $p(u_{i+1}, u_j)$ parallel edges from u_{i+1} to u_j with $p(u_{i+1}, u_j)$ the number of $\mathcal{P}$ paths associated with this pair. Hence, we have constructed a 4−layered subgraph as in Lemma 2.1 and this lemma implies that $R(L_i, L_j) + R(L_{i+1}, L_{j+1}) \geq k$. The statement follows by summing the contribution of all such pairs. ∎

We use the following special case of the Max-Flow Min-Cut theorem (c.f., [3]).

Theorem 2.3 ***(Menger's Theorem).*** *Let $s, t \in G$. The maximum number of edge-disjoint (s,t)−paths in G equals the size of a minimum (s,t)−cut.*

Lemma 2.4. *Let s, t be a pair of nodes in a simple graph G with $d_G(s,t) \geq 4p\log n + 2$. Then there exists an (s,t)-cut C so that $R(G) - R(G - C) = \Omega(|C|p^2)$.*

Proof. Consider the corresponding $d_G(s,t)$ BFS layers from s to t, where nodes that cannot reach t are deleted. Let X_i be the layer at distance i from s, and let Y_i be the layer at distance i to t. Let k_j be the maximum number of edge-disjoint $(X_{j\cdot p}, Y_{j\cdot p})$-paths in the graph G_j induced by all the layers starting with $X_{j\cdot p}$ and ending at $Y_{j\cdot p}, j = 1, \ldots, 2\log n$.

We claim that there exists an index j with $k_j \leq 2 \cdot k_{j-1}$. Otherwise, because $k_0 \geq 1$, we have $k_j \geq 2^j$. For $j = 2\log n + 1$ we get $k_j \geq 2n^2$, which is not possible in a simple graph.

Let j be such an index with $k_j \leq 2 \cdot k_{j-1}$, and let C be a minimum $(X_{j\cdot p}, Y_{j\cdot p})$-cut, so $|C| = k_j$. We now apply Lemma 2.2 on the graph G_{j-1}. Note that G_{j-1} contains at least $|C|/2$ edge-disjoint paths between its first layer $X_{(j-1)\cdot \ell}$ and its last layer $Y_{(j-1)\cdot \ell}$; this is because $k_j = |C|$ by Menger's Theorem, and $k_{j-1} \geq k_j/2$ by the choice of j. Because C separates the first and the last p layers of G_{j-1}, the statement follows from Lemma 2.2. ∎

Corollary 2.5. *For UDM there exists an algorithm that for any integer $\ell > 4\log n + 2$ finds in $\tilde{O}(mn^2/\ell^2)$ time a K-multicut B with $|B| = O((n\log n)^2/\ell^2)$, where $K = \{(u,v) : d(u,v) \geq \ell\}$.*

Proof. Let $p = \ell/(4\log n + 2)$. The algorithm starts with $B = \emptyset$. Although there is an (s,t)-path for some $(s,t) \in K$ it computes an (s,t)-cut $C = C_{st}$ as in Lemma 2.4, and sets $B \leftarrow B \cup C$, $G \leftarrow G - C$. We claim that at the end of the algorithm $|B| = O(R(G)/p^2) = O(n^2/p^2)$; we get that $|B| = O((n\log n)^2/\ell^2)$ by substituting $p = \ell/(4\log n)$. Lemma 2.2 implies that there exists a constant $\alpha > 0$ so that each time C_{st} is deleted, $R(G)$ is reduced by at least $\alpha|C_{st}|p^2$. Thus, we get

$$\alpha p^2 \cdot |B| \leq \alpha p^2 \cdot \sum_{(s,t)\in K} |C_{st}| \leq R(G) \leq n^2. \qquad (1)$$

The dominating time at each iteration is spent for computing a cut as in Lemma 2.4. This can be done using $O(\log n)$ max-flow computations as follows directly from the proof of Lemma 2.4. Thus, it can be computed in $\tilde{O}(m|C_{st}|)$ time using the Ford-Fulkerson algorithm [6]. Thus, the total time required is $\tilde{O}(m|B|) = \tilde{O}(mn^2/\ell^2)$. ∎

We are now ready to prove Theorem 1.1. Given an integer ℓ, apply the following procedure starting with $A, B = \emptyset$:

Phase 1:

While there is an (s,t)-path P with $|P| \leq \ell$ for some $(s,t) \in K$ do:

$A \leftarrow A + P, G \leftarrow G - P.$

End While

Phase 2: Find in $G - A$ a K-multicut B as in Corollary 2.5.

For any integer ℓ, the algorithm computes a K-multicut $C = A \cup B$ of size $\ell \cdot opt + O((n\log n)^2/\ell^2)$; $|A| \leq \ell \cdot opt$ because any K-multicut contains at least one edge of each path removed, and $|B| = O((n\log n)^2/\ell^2)$ by Corollary 2.5. As was explained in the introduction, we execute the algorithm for $\ell = 1, \ldots, (n\log n)^{2/3}$, and among the multicuts computed output one of minimum size. For $\ell = (n\log n)^{2/3}/opt^{1/3}$ we get the claimed approximation ratio.

Let us now discuss the implementation of the algorithm. After executing Phase 1 at iteration ℓ, the graph $G - A$ is used as an input for iteration $\ell + 1$. As the total length of the paths removed is at most n^2, and each iteration requires a shortest path computation the total time of Phase 1 executions is $O(mn^2)$. The total time of Phase 2 executions is $\tilde{O}\big(\sum_{i=1}^{n^{2/3}} mn^2/i^2\big) = \tilde{O}(mn^2)$. Thus, the time complexity is as claimed, and the proof of Theorem 1.1 is complete.

3. AN $O(n^{2/3})$-APPROXIMATION ALGORITHM FOR DM

3.1. The Algorithm:

Consider the following algorithm:

Input: An instance (G, u, K) of DM, and integers ℓ, μ.
Initialization: $C \leftarrow \emptyset$.
While in G there is an (s,t)-path P for some $(s,t) \in K$ do:

(a) Let P' be the union of the first and the last ℓ edges of P ($P' = P$ if $|P| < 2\ell$);
(b) Among the (s,t)-cuts in G disjoint with P' compute one C' of minimum capacity ($u(C') = \infty$ if $P' = P$);

```
(c) If $P = P'$ then $C \leftarrow C \cup P, G \leftarrow G - P$.
    (c i) Else, if $u(C') > \mu$ then: $u_e \leftarrow u_e - \min\{u_e : e \in P'\}$ for every $e \in P'$; $C \leftarrow C \cup P'_0, G \leftarrow G - P'_0$, where $P'_0 = \{e \in P' : u_e = 0\}$.
    (c ii) Else ($u(C') \leq \mu$) $C \leftarrow C \cup C', G \leftarrow G - C'$.
End While
```

Lemma 3.1. *At the end of the algorithm C is a K-multicut. If $\mu \geq opt$ then $u(C) \leq \mu \cdot (2\ell + n^2/\ell^2)$.*

Proof. Assume that $\mu \geq opt$. Consider a specific iteration of the main loop, and the edge sets P', C' found. There are three possible cases.

If $P' = P$, then P is added to C. Because the optimum contains at least one of these edges, the number of edges added in these case throughout the algorithm is at most $2 \cdot \ell \cdot opt \leq 2 \cdot \ell \cdot \mu$. If $u(C') > \mu$, then $u(C') > \mu \geq opt$. This implies that *any* minimum K-multicut contains at least one edge from P'. Hence, after setting $u_e \leftarrow u_e - \min\{u_e : e \in P'\}$ for every $e \in P'$ the optimum decreases by at least $\min\{u_e : e \in P'\}$. Because $|P'| = 2\ell$, the total capacity of the edges in all sets P'_0 added into C during the algorithm is at most $2\ell opt \leq 2\ell\mu$.

Otherwise, if $u(C') \leq \mu$ then $R(G) - R(G - C') \geq \ell^2$. Thus the total *number* of cuts C' removed during the algorithm $\leq n^2/\ell^2$, and their total capacity $\leq \mu n^2/\ell^2$.

To see that $R(G) - R(G - C') \geq \ell^2$, let P'_F and P'_L be the first and the last ℓ nodes in P, respectively. We claim that $R_G(P'_F, P'_L) = |P'_F| \cdot |P'_L| = \ell^2$ and $R_{G-C'}(P'_F, P'_L) = 0$. The first statement follows from the simple observation that P'_F, P'_L belong to the same path P of G, and thus $d_G(u, v) < \infty$ for every pair u, v with $u \in P'_F, v \in P'_L$. To see the second statement, note that $d_{G-C'}(u, v) = \infty$ for every such pair u, v, as otherwise there would be an (s, t)-path in $G - C'$, contradicting that C' is an (s, t)-cut in G. ∎

We are now ready to prove Theorem 1.2. As was mentioned in the Introduction, for $\ell = n^{2/3}$ we use binary search to find the minimum integer μ so that a multicut of capacity $\leq \mu \cdot (2\ell + n^2/\ell^2)$ is returned. Lemma 3.1 implies that $\mu \leq opt$, and the required ratio follows.

We now analyze the running time. We can assume that $u_e \in \{1, \ldots, n^4\}$ or $u_e = \infty$ for every $e \in E$. In this case binary search for the appropriate μ requires $O(\log(n^4)) = O(\log n)$ iterations. Indeed, let c be the least integer so that $\{e \in E : u_e \leq c\}$ is a K-multicut. Edges of capacity $\geq cn^2$ do not belong to any optimal solution, and their capacity is set to ∞. Edges of capacity $\leq c/n^2$ are removed, as adding all of them to the solution affects only the constant in the approximation ratio. This gives an instance with $u_{\max}/u_{\min} \leq n^4$, where $u_{\max}$ and $u_{\min}$ denote the maximum finite and the minimum nonzero capacity of an edge in E, respectively. Further, for every $e \in E$ set $u_e \leftarrow \lceil u_e/u_{\min} \rceil$. It is easy to see that the loss incurred in the approximation ratio is only a constant, which is negligible in our context.

The dominating time is spent for computing $O(m)$ minimum cuts at step (b); each such computation leads to a removal of an edge, because reducing the capacities along P' by $\min\{u_e : e \in P'\}$ guarantees that at least one edge gets capacity zero. As a max-flow/min-cut computation can be done in $\tilde{O}(nm)$ time (c.f., [2]), the total running time is $\tilde{O}(nm^2)$. This finishes the proof of Theorem 1.2.

Acknowledgment

The second author thanks Joseph Cheriyan for suggesting the problem and for helpful discussions, and Howard Karloff and Aravind Srinivasan for useful discussions.

REFERENCES

[1] J. Cheriyan, H. Karloff, and Y. Rabani, Approximating directed multicuts, Combinatorica, to appear.

[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to algorithms, 2nd ed., MIT Press, Cambridge, MA; 1998.

[3] S. Even, Graph algorithms, Computer Science Press, Haifa, 1979.

[4] G. Even, S. Naor, B. Schieber, and S. Rao, Divide-and-conquer approximation algorithms via spreading metrics, ACM 47(2000), 585–616.

[5] L. Fleischer, Approximating fractional multicommodity flows independent of the number of commodities, SIAM J Discrete Math 13(2000), 505–520.

[6] L.R. Ford and D.R. Fulkerson. Flows in networks, Princeton University Press, Princeton, NJ, 1962.

[7] N. Garg, V. Vazirani, and M. Yannakakis, Approximate max-flow min-(multi)cut theorems and their applications, SIAM Comput 25(1996), 235–251.

[8] N. Garg, V. Vazirani, and M. Yannakakis, Primal-dual approximation algorithms for integral flow and multicut in trees, Algorithmica 18(1997), 3–20.

[9] A. Gupta, Improved approximation algorithm for directed multicut, Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2003, pp. 12–14.

[10] D.S. Hochbaum (editor), Approximation algorithms for NP-hard problems, PWS Publishing Company, Boston, MA, 1997.

[11] P.N. Klein, S.A. Plotkin, S. Rao, and E. Tardos, Approximation algorithms for Steiner and directed multicuts, J. Algorithms 22(1997), 241–269.

[12] P.D. Seymour, Packing directed circuits fractionally, Combinatorica 15(1995), 281–288.

[13] K. Varadarajan and G. Venkataraman, Graph decomposition and a greedy algorithm for edge-disjoint paths, Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2004, pp. 379–380.