The maximum integer multiterminal flow problem in directed graphs

Cédric Bentz*

Abstract

Given an arc-capacitated digraph and k *terminal* vertices, the *directed maximum integer multiterminal flow* problem is to route the maximum number of flow units between the terminals. We introduce a new parameter $k_L \leq k$ for this problem and study its complexity with respect to k_L.

Keywords: integer multiterminal flow, directed graphs, APX-hardness, approximation algorithms.

1 Introduction

Routing problems in networks are commonly modelled by flow or multicommodity flow problems. Given an edge-capacitated graph (directed or undirected), the goal is to route flow units (requests) between prespecified vertices. When one seeks to route the maximum number of flow units from a unique source to a unique sink, the problem is the famous *maximum flow problem*. The Ford-Fulkerson's theorem [9] gives a good characterization for this case, which is efficiently solvable [1]. In particular, this theorem states that, if the capacities are integral, the value of a maximum integer flow is equal to the value of a minimum cut, i.e., to the value of a minimum weight set of edges whose removal separates the source from the sink. Unfortunately, this does not hold for more general variants.

One of the most studied variant is the *maximum integer multiflow problem*: given an edge-capacitated graph $G = (V, E)$ and a list of source-sink pairs, the goal is to simultaneously route the maximum number of flow units, each unit being routed from one source to its corresponding sink, while respecting the capacity constraints on the edges. For two source-sink pairs, this problem is strongly $\mathcal{NP}$-hard both in directed acyclic graphs and in undirected graphs [8] (more precisely, given an unweighted undirected or directed acyclic graph, two pairs (s_1, s'_1) and (s_2, s'_2) and a (polynomially

*LRI, Université Paris-Sud and CNRS, 91405 Orsay Cedex, France (e-mail: cedric.bentz@lri.fr)

bounded) demand d, it is $\mathcal{NP}$-complete to decide whether we can simultaneously route $d+1$ disjoint paths, d paths from s_1 to s_1' and one from s_2 to s_2'), and even APX-hard in undirected and directed graphs [16, p. 489] (implying that there is no PTAS if $\mathcal{P} \neq \mathcal{NP}$; see Section 2 for a definition). Moreover, if $\mathcal{P} \neq \mathcal{NP}$, it cannot be approximated efficiently within $|E|^{\frac{1}{2}-\epsilon}$ in digraphs [16] and within $(\log |E|)^{\frac{1}{3}-\varepsilon}$ in undirected graphs [2], for any $\epsilon > 0$ (recall that, for a maximization (resp. minimization) problem, an α-approximation algorithm is a polynomial-time algorithm that always outputs a feasible solution whose value is at least $1/\alpha$ times (resp. at most α times) the value of an optimal solution). On the positive side, an $O\left(\sqrt{|E|}\right)$-approximation algorithm is known for general digraphs [18], and an $O\left(\sqrt{|V|}\right)$-approximation is known for directed acyclic graphs [4, 20] and undirected graphs [4]. For further results on the tractability and approximability of special cases, see [6, 11].

The corresponding generalization of the problem of finding a minimum cut is the *minimum multicut problem*, which asks to select a minimum weight set of edges whose removal separates each source from its corresponding sink. This problem is also $\mathcal{NP}$-hard (even in very restricted classes of graphs, such as unweighted stars [15]), and has a noticeable relationship with the former: the continuous relaxations of the linear programming formulations of the two problems are dual [14]. In particular, this interesting property has been used to design good approximation algorithms for both problems [14, 15, 22]. Further results and references concerning the maximum integer multiflow and minimum multicut problems can be found in [1, 6, 22].

Another generalization of the maximum flow problem is the *maximum integer multiterminal flow problem* (MaxIMTF): given an edge-capacitated graph and a set $T = \{t_1, \ldots, t_k\}$ of *terminal* vertices, MaxIMTF is to route the maximum number of flow units between the terminals. Note that this problem is a particular maximum integer multiflow problem in which the source-sink pairs are (t_i, t_j) for $i \neq j$. The associated *minimum multiterminal cut problem* (MinMTC) is to select a minimum weight set of edges whose removal separates t_i from t_j for $i \neq j$. Note that MaxIMTF and MinMTC also have the duality relationship mentioned above. MinMTC has been widely studied in the undirected case [5, 6, 7, 23], and the directed case has also received some attention: Garg et al. [13] show that it is $\mathcal{NP}$-hard even for $k = 2$ and give an $2\log_2 k$-approximation algorithm, and Naor and Zosin [19] give a 2-approximation algorithm. However, the algorithm of Garg et al. has an interesting property: it computes a multiterminal cut whose value is at most $2\log_2 k$ times the value of an integer multiterminal flow, and hence is an $2\log_2 k$-approximation for both MinMTC *and* MaxIMTF (while the algorithm of Naor and Zosin does not provide an approximate solution for MaxIMTF). Costa et al. [6] show that MaxIMTF and MinMTC are polynomial-time solvable in acyclic directed graphs by using a simple reduction to a maximum flow and a minimum cut problem, respectively. To the best of our knowledge, these are the only results about MaxIMTF in directed graphs. In undirected graphs, MaxIMTF has recently be shown to be polynomial-time solvable via the ellipsoid method [17]: the resolution is based on the related Mader's theorem on T-paths [21, Chap. 73]. Algorithmic aspects of special cases have also been studied (see [3] and [12]). However, it can be noticed that, for all the problems mentioned above, the general directed case is "harder" than the undirected one, since there exists a linear reduction from the latter to the former: simply replace each edge by the gadget given in [21, (70.9) on p. 1224].

The motivation of this paper is to explore further the complexity of MaxIMTF in directed graphs. We say that a terminal is *lonely* if it lies on at least one directed cycle containing no other terminal, and we let T_L denote the set of lonely terminals and $k_L = |T_L|$ (note that $k_L = 0$ in directed acyclic graphs). We shall see that k_L is a particularly interesting parameter for better understanding the complexity and approximability of MaxIMTF in digraphs (k_L is small when either k or the number of directed cycles is small, and both parameters can be arbitrarily larger than k_L): this paper gives a complete classification of the tractable and intractable cases of MaxIMTF in digraphs with respect to this parameter. Moreover, some of our results will extend to MinMTC.

Intuitively, the condition $k_L = 0$ ensures that any flow unit routed from a terminal t_i to one of the k terminals (including t_i itself) will go through at least one terminal different from t_i; therefore, it can be assumed to be routed from one terminal to another (different) terminal.

We first show that MaxIMTF is $\mathcal{NP}$-hard to approximate within $2-\epsilon$ for any $\epsilon > 0$ in directed graphs, even if $k_L = k = 2$ or if $k_L = 1$ and $k = 3$ (Section 2). Then, we prove MaxIMTF to be tractable when $k_L = 0$ by reducing it to a simple maximum flow problem, and improve the $2\log_2 k$-approximation algorithm of Garg et al. [13] by providing an $2\log_2(k_L + 2)$-approximation algorithm for the general case (Section 3). We also show the tightness of our analysis. Eventually, we show the case $k_L = 1$ and $k = 2$ to be polynomial-time solvable (Section 4). We leave as open the problem of deciding whether there exists an $O(1)$-approximation algorithm for MaxIMTF in digraphs.

Note that, throughout this paper, we consider only *simple* graphs. We call *Directed* MaxIMTF the problem MaxIMTF defined in directed graphs.

2 APX-hardness proof

We show in this section that Directed MaxIMTF is APX-hard (i.e., there exists an $\alpha > 1$ such that there is no α-approximation algorithm for this problem), even if $k = k_L = 2$ (or $k_L = 1$ and $k = 3$). First, notice that, when $k = 2$ (i.e., $T = \{t_1, t_2\}$), Directed MaxIMTF is *equivalent* (in digraphs) to the maximum integer multiflow problem with two source-sink pairs (s_1, s'_1) and (s_2, s'_2). Indeed, given (s_1, s'_1) and (s_2, s'_2), we can obtain an equivalent instance of Directed MaxIMTF by defining two new terminals, t_1 and t_2, and by linking (by arcs with sufficiently large capacities) t_1 to s_1, s'_2 to t_1, t_2 to s_2 and s'_1 to t_2: any flow unit routed between t_1 and t_2 is either routed from s_1 to s'_1 or from s_2 to s'_2. Conversely, given two terminals t_1 and t_2, we can define $s_1 = t_1$, $s'_1 = t_2$, $s_2 = t_2$ and $s'_2 = t_1$. Obviously, this transformation does not apply to the undirected case.

We will use the fact that, if $\mathcal{P} \neq \mathcal{NP}$, there is no ρ-approximation algorithm for the maximum integer multiflow problem in digraphs even with two source-sink pairs, for any $\rho < 2$ (recall that this problem is $APX-hard$ even in undirected graphs [16, p. 489]). Indeed, the problem $P2$: *given a digraph G and two vertex pairs (s_1, s'_1) and (s_2, s'_2), decide whether there simultaneously exist in G two paths, one from s_1 to s'_1, and the other from s_2 to s'_2* is $\mathcal{NP}$-complete [10] (an instance of $P2$ is called *feasible* if two such paths exist). This result was used in [16] to prove a strong inapproximability result for the maximum integer multiflow problem in digraphs.

Now, let $\mathcal{I}$ be any instance of $P2$. If there exists an approximation algorithm for the maximum integer multiflow problem with a ratio $\rho < 2$, such an algorithm would output a solution of value 1 if $\mathcal{I}$ is not feasible, and of value 2 otherwise (since $\frac{2}{\rho} > 1$): this would give a polynomial-time algorithm for solving $P2$. This implies:

Theorem 1. Directed MaxIMTF *is $\mathcal{NP}$-hard to approximate within $2 - \epsilon$ for any $\epsilon > 0$, even when $k_L = k = 2$.*

Note that this lower bound is tight for $k = 2$, since applying Garg et al.'s algorithm yields a 2-approximation in this case [13]. Moreover, this result essentially matches the complexity result (namely, APX-hardness) for the associated cut problem MinMTC in directed graphs [13]. Now, we prove the same result for the case $k = 3$, $k_L = 1$. The proof is quite similar: given two vertex pairs (s_1, s'_1) and (s_2, s'_2), define three new terminals t_1, t_2, t_3, and add four arcs, $(t_1, s_1), (s'_2, t_1), (t_2, s_2), (s'_1, t_3)$; then, any path leaving t_1 is routed towards t_3. Note that $T_L = \{t_1\}$, since the only terminal lying on a directed cycle is t_1. Hence:

Theorem 2. Directed MaxIMTF *is $\mathcal{NP}$-hard to approximate within $2 - \epsilon$ for any $\epsilon > 0$, even when $k = 3$ and $k_L = 1$.*

We shall deal with the case $k_L = 0$ (which generalizes the acyclic case) in Section 3 and with the case $k_L = 1$ and $k = 2$ in Section 4.

3 Exact and approximation algorithms

From the previous section, Directed MAXIMTF is APX-hard for $k_L \geq 1$. Hence, if $\mathcal{P} \neq \mathcal{NP}$, the only efficient algorithms one can expect to design are approximation algorithms. In this section, we improve the $2\log_2 k$-approximation algorithm of Garg et al. [13] and give an $2\log_2(k_L + 2)$-approximation algorithm for Directed MAXIMTF.

The basic idea of our algorithm is to combine the algorithm of Garg et al. with an improvement of an idea used in [6, Proposition 3]. The main idea of the proof of [6, Proposition 3] (that shows that MAXIMTF and MINMTC are polynomial-time solvable in directed acyclic graphs) is to split up each terminal vertex t_i into two new vertices, t'_i and t''_i, such that all the vertices in $\Gamma^-(t_i)$ are linked to t'_i and t''_i is linked only to the vertices in $\Gamma^+(t_i)$ (where, for a digraph $G = (V, A)$ and $v \in V$, $\Gamma^+(v) = \{u \in V$ such that $(v, u) \in A\}$ and $\Gamma^-(v) = \{u \in V$ such that $(u, v) \in A\}$). Then, we add two new vertices, σ and τ, and link (by arcs with large capacities) every t'_i to τ and σ to every t''_i. Finally, we compute a maximum flow between σ and τ (obviously, we assume that the capacities are integral). The obtained flow is a valid integer multiterminal flow for the initial instance if, in the modified instance, no flow unit is routed from t''_i to t'_i for some i.

We can obtain an interesting strengthening of [6, Proposition 3] by noticing that, if there is no lonely terminal, then, by splitting up the terminals as explained, there will remain no directed path from t''_i to t'_i for each i, and hence we can solve MAXIMTF and MINMTC using the above technique.

Theorem 3. MINMTC *and* MAXIMTF *are polynomial-time solvable in directed graphs if $k_L = 0$, by using a max flow-min cut algorithm.*

Actually, if we want to guarantee that, after splitting up each terminal, the modified graph does not admit a directed path from t''_i to t'_i for some i (otherwise, we cannot be sure that the flow we will compute in the modified graph will be a valid multiterminal flow in the initial graph), this is essentially the best (i.e., weakest) assumption that can be made.

Theorem 4. *After splitting up all the terminals, there is no directed path between t''_i and t'_i for each i if and only if $k_L = 0$.*

Proof. Follows directly from the definition of T_L and the way we split up the terminals. □

Theorems 3 and 4 show the importance of the parameter k_L for both MAXIMTF and MINMTC. Moreover, this suggests the following approach (algorithm $\mathcal{A}$) for finding approximate solutions for these two problems:

1. If $k \geq k_L + 2$, split up each terminal $t_i \in T - T_L$ into t'_i and t''_i as explained above, add the two vertices σ and τ, and link (by arcs with sufficiently large capacities) every t'_i to τ and σ to every t''_i;

2. Compute a solution for this new instance (i.e., where the terminal set is $T_L \bigcup \{\sigma, \tau\}$) by using the algorithm of Garg et al. [13].

Hence, the main difference with the algorithm in [13] is that, before using their divide-and-conquer strategy, we "replace" the terminals in $T \setminus T_L$ by only two terminals, σ and τ. This implies that we use Garg et al.'s algorithm on an instance with $k_L + 2$ terminals, and so we obtain an approximation factor of $2\log_2(k_L + 2)$ (instead of $2\log_2 k$). The key point is that any flow unit routed during the algorithm from σ to τ via t''_i and t'_i for some i with $t_i \notin T_L$ (if any) can be re-routed either between σ and a terminal in T_L or between a terminal in T_L and τ (because $t_i \in T - T_L$). This implies:

Theorem 5. *Algorithm $\mathcal{A}$ is an $2\log_2(\min(k_L + 2, k))$-approximation algorithm for* MaxIMTF *in directed graphs.*

Moreover, the above transformation can be of independent interest, since it can always be applied in order to reduce the instance size (i.e., to reduce the number of terminals from k to $k_L + 2$) in any procedure computing an integer multiterminal flow (e.g., an exact implicit enumeration procedure).

Actually, we can prove that our analysis of the approximation ratio of algorithm $\mathcal{A}$ is tight. To do this, we use an instance built on an undirected tree with 2^p vertices. We are given a path $P = u_0, u_1, \ldots, u_p$, and, for each $i \in \{0, \ldots, p-1\}$, $2^i - 1$ leaves are linked by an edge of capacity 1 to u_i (let this set of leaves be $\mathcal{L}_i$). All the vertices are terminal vertices (and hence $k = 2^p$, i.e, $p = \log_2 k$), and the p edges of P are valued by a big integer N. To transform this undirected graph into a directed one, we replace each edge by the gadget given in [21, (70.9) on p. 1224]: each arc of the gadget has the capacity of the initial edge. In this instance, all the terminals are lonely (i.e., $k = k_L$), and hence our algorithm simply consists in applying Garg et al.'s algorithm. Hence, we will prove the tightness of their analysis, and this will imply the tightness of ours. We assume without loss of generality that Garg et al.'s algorithm always breaks ties in the worst possible way, and that it computes a solution by iteratively separating u_{p-i+1} and $\mathcal{L}_{p-i}$ from u_{p-i}: indeed, this will result in a binary search on k. Moreover, the first max flow computed at the i^{th} iteration will consist in routing N units of flow between u_0 and u_{p-i+1} and one unit of flow between u_{p-i} and each leaf in $\mathcal{L}_{p-i}$ (the second max flow is symmetric; thus, it has the same value). The min cut will consist in cutting two different arcs (adjacent to the same terminal) on the gadget linking u_{p-i} to u_{p-i+1} and on the gadget linking u_{p-i} to each leaf in $\mathcal{L}_{p-i}$. The algorithm of Garg et al. outputs a flow of value at most $N+(2^{p-1}-1)+(2^{p-2}-1)+\cdots+(2^1-1) = N+2^p-p-1$ (even if it combines flows obtained in different iterations in the best possible way) and a cut of value $2(pN + 2^p - p - 1)$. The ratio between these two values is equal to $\frac{2(pN+2^p-p-1)}{N+2^p-p-1}$, and tends to $2p = 2\log_2 k_L$ as N increases, establishing the tightness of Garg et al.'s analysis. However, this does not imply that one cannot hope for a better approximation ratio by using a different algorithm.

Eventually, note that tighter ratios can be obtained for special cases (for example, when $k_L = 1$, our analysis yields a 2-approximation).

4 $k_L = 1$ and $k = 2$: A tractable case

Directed MaxIMTF is APX-hard even when $k = k_L = 2$ and when $k_L = 1$ and $k = 3$ (see Section 2), and polynomial-time solvable when $k_L = 0$ (see Section 3). In this section, we settle the last case and show that Directed MaxIMTF is polynomial-time solvable when $k_L = 1$ and $k = 2$.

Let $T = \{t_1, t_2\}$ and $T_L = \{t_1\}$. From the definition of T_L, there exists at least one directed cycle containing t_1 but not t_2, and there exists no directed cycle containing t_2 but not t_1. We are going to show that, in fact, this instance can be transformed into an equivalent one in which $k_L = 0$; then, we will conclude by using Theorem 3. We show the following lemma:

Lemma 1. *Let $\mathcal{I}$ be an instance of* Directed MaxIMTF *with $T = \{t_1, t_2\}$ and $T_L = \{t_1\}$. On any directed cycle containing t_1 but not t_2, there is a* removable *arc, i.e., an arc not used by at least one optimal solution for $\mathcal{I}$.*

Proof. Let $\mathcal{C} = \{t_1, u_1, u_2, \ldots, u_p, t_1\}$ be a directed cycle not containing t_2. We show that, on $\mathcal{C}$, there exists an arc lying neither on an elementary path from t_1 to t_2 nor on an elementary path from t_2 to t_1. Let i be such that there is a path from u_i to t_2 not containing t_1, but there is no path from u_{i+1} to t_2 not containing t_1. If i exists, then the arc (u_i, u_{i+1}) can be removed. Indeed, it cannot lie on an elementary path from t_1 to t_2 (since there is no path from u_{i+1} to t_2 not containing t_1), and it cannot lie on an elementary path from t_2 to t_1 (since otherwise there is a path from t_2 to u_i not containing t_1, and $t_2 \in T_L$). If i does not exist, then we can remove either (t_1, u_1) (if there is no path from u_1 to t_2 not containing t_1) or (u_p, t_1) (if there is a path from u_p to t_2 not containing t_1). Lemma 1 follows. $\square$

The proof of Lemma 1 also provides an algorithm to solve MaxIMTF in this case, by iteratively removing arcs until there remains no lonely terminal.

Acknowledgments

The author thanks the anonymous referees for their numerous remarks, that greatly helped improving the content and the presentation of the present paper.

References

[1] A.K. Ahuja, T.L. Magnanti and J.B. Orlin. Network Flows – Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs, New Jersey (1993).

[2] M. Andrews and L. Zhang. Hardness of the undirected edge-disjoint paths problem. Proceedings STOC'05 (2005).

[3] A. Billionnet and M.-C. Costa. Multiway cut and integer flow problems in trees. Proceedings CTW04, ENDM **17** (2004) 105–109.

[4] C. Chekuri, S. Khanna and F.B. Shepherd. An $O(\sqrt{n})$-approximation and integrality gap for EDP and UFP in undirected graphs and DAGs. Working paper, submitted (2005).

[5] D.Z. Chen and X. Wu. Efficient algorithms for k-terminal cuts on planar graphs. Algorithmica **38** (2004) 299–316.

[6] M.-C. Costa, L. Létocart and F. Roupin. Minimal multicut and maximal integer multiflow: a survey. Eur. J. Op. Res. **162** (2005) 55–69.

[7] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour and M. Yannakakis. The complexity of multiterminal cuts. SIAM J. Computing **23** (1994) 864–894.

[8] S. Even, A. Itai and A. Shamir. On the complexity of timetable and multicommodity flow problems. SIAM J. Computing **5** (1976) 691–703.

[9] L.R. Ford and D.R. Fulkerson. Maximal Flow Through a Network. Canadian Journal of Mathematics **8** (1956) 339–404.

[10] S. Fortune, J. Hopcroft and J. Wyllie. The directed subgraph homeomorphism problem. Theoretical Computer Science **10** (1980) 111–121.

[11] A. Frank. Packing paths, circuits and cuts-a survey. In B. Korte, L. Lovász, H.J. Prömel and A. Schrijver (eds). *Paths, Flows and VLSI-Layout.* Algorithms and combinatorics **9** (1990) 47–100. Springer.

[12] A. Frank, A. Karzanov and A. Sebö. On integer multiflow maximization, SIAM J. Discrete Mathematics **10** (1997) 158–170.

[13] N. Garg, V.V. Vazirani and M. Yannakakis. Multiway cuts in directed and node weighted graphs. Proceedings ICALP'94 (1994) 487–498.

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

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

[16] V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd and M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. Journal of Computer and System Sciences **67** (2003) 473–496.

[17] J.C.M. Keijsper, R.A. Pendavingh and L. Stougie. A linear programming formulation of Mader's edge-disjoint paths problem. Journal of Combinatorial Theory, Series B **96** (2006) 159–163.

[18] J. Kleinberg. Approximation algorithms for disjoint paths problems. PhD thesis, MIT, Cambridge, MA (1996).

[19] J. Naor and L. Zosin. A 2-approximation algorithm for the directed multiway cut problem. SIAM J. Computing **31** (2001) 477–482.

[20] T. Nguyen. On the Disjoint Paths Problem. Submitted, October 2005.

[21] A. Schrijver. Combinatorial Optimization - Polyhedra and Efficiency. Algorithms and Combinatorics 24 (2003). Springer.

[22] V.V. Vazirani. *Approximation algorithms* (2001). Springer-Verlag.

[23] W.-C. Yeh. A Simple Algorithm for the Planar Multiway Cut Problem. Journal of Algorithms **39** (2001) 68–77.