Minimum Cost Integral Network Coding

Tao Cui and Tracey Ho
Department of Electrical Engineering
California Institute of Technology
Pasadena, CA, USA 91125
Email: {taocui, tho}@caltech.edu

***Abstract*— In this paper, we consider finding a minimum cost multicast subgraph with network coding, where the rate to inject packets on each link is constrained to be integral. In the current minimum cost network coding formulation, the optimal solution cannot always be integral. Fractional rates can be well approximated by choosing the time unit large enough, but this increases the encoding and decoding complexity as well as delay at the terminals. We formulate this problem as an integer program, which is shown to be NP-hard. A greedy algorithm and an algorithm based on linear programming rounding are proposed, both of which can be decentralized. We show by simulation that our algorithms offer good approximation ratios on random graphs.**

I. Introduction

A fundamental problem in network design is how to increase the amount of information transferred by the network. In today's packet networks, each node's functions are limited to the forwarding or replication of received packets. However, by using network coding, where each node is allowed to perform algebraic operations on received packets, it has been shown that the ability of the network to transfer information can be significantly improved [1], [2].

The concept of network coding was introduced in a seminal paper by Ahlswede *et. al.* [1] and has attracted significant interest from various research communities. It was shown in [1] that the capacity of the network is equal to the size of the minimum cut that separates the source and any terminal. In a subsequent work, Li *et. al.* [2] proved that linear network codes are sufficient to achieve the capacity of the network. An algebraic framework for linear network codes on directed graphs is developed by Koetter and Médard [3]. This framework was used by Ho *et. al.* [4] to construct random distributed network coding, which achieves the network capacity with probability exponentially approaching 1 with the code length. All these papers assume that network resources have already been dedicated. The goal is to achieve the maximum rate that the information can be transferred through the network.

On the other hand, decentralized methods for determining the allocation of network resources to a particular network coded connection on directed graphs were proposed by Lun *et. al.* [5]. The decentralized methods in [5] generate the subgraph, which assigns the rate to inject packets on each link. The network codes designed in [2]–[4] are then applied to code over the subgraph. In the network coding literature [1]–[4], the capacity of any link is assumed to be unity or more generally, an integer. It might seem that the decentralized minimum-cost multicast problem is completely solved by using the decentralized subgraph generation methods in [5] and the random distributed network coding in [4]. However, the optimal solution given by [5] cannot always be integral. As remarked in [3], fractional capacities can be well approximated by choosing the time unit large enough. But, such capacity scaling will increase the encoding and decoding complexity, and introduce a large delay at the receiver. All these factors will limit the application of network coding in delay sensitive or computational resources limited network applications.

In this paper, we consider finding a minimum cost multicast subgraph with network coding on directed graphs, where the packet transmission rate on each link is constrained to be integral. Our problem is a generalization of the Steiner tree problem (STP) [6]. The STP is known to be NP-hard [6]. Various approximation algorithms for the STP are available in the literature. For the undirected Steiner tree problem, a heuristic algorithm is given in [7] based on Prim's minimum spanning tree algorithm. A factor 2 approximation algorithm is given in [8] by using linear programming (LP) rounding techniques. On the other hand, the directed Steiner tree problem is usually much harder to solve. A heuristic algorithm is given in [9] by extending the results in [7]. Wong [10] gave a primal-dual algorithm for directed Steiner tree problem.

Jain's LP rounding algorithm can also be applied to give a 2-approximation algorithm for the generalized Steiner network problem on undirected graphs where given requirements $r_{i,j}$ for each pair of vertices i,j we seek to find a minimum cost subgraph with $r_{i,j}$ edge-disjoint paths between i and j. We consider directed graphs, with requirements $r_{s,t}{=}h$ where h is an integer, s is a given source node and t is one of a set of sink nodes. We show that the IP with convex and separable cost can be transformed into an equivalent integer linear program with linear cost. By reducing set cover to our problem, we show that the solution by [5] cannot be always integral. By generalizing the Prim's algorithm for find minimum spanning tree, a greedy algorithm is proposed, which has an approximation factor k. We give a bad example that the approximation ratio k is indeed achieved by the greedy algorithm. A new LP rounding scheme is also proposed. When used together with capacity scaling, it gives a tradeoff between encoding and decoding complexity, delay and approximation ratio. Moreover, the proposed approximation algorithms can be readily decentralized. Simulation results show that our

This work has been supported in part by DARPA grant N66001-06-C-2020, Caltech's Lee Center for Advanced Networking and a gift from Microsoft Research.

algorithms offer good approximation ratios on random graphs.

II. Model, Background and Motivation

A. Network Coding

The communication network is represented by a directed graph $\mathcal{G}=(\mathcal{N},\mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of arcs in $\mathcal{G}$. Each arc (i,j) represents a lossless point-to-point link from node i to node j. The capacity $c_{i,j}$ of arc (i,j) is defined to be the number of packets that can be sent over (i,j) in one time unit. We assume that arc capacities are non-negative integer numbers. We consider single session multicast in this paper, where a source node $s\in\mathcal{N}$ must transmit h packets per unit time to every terminal in a set of k terminals $\mathcal{T}\subset\mathcal{N}$. We assume that h is an integer in this paper, which arises naturally in practice.

In network coding, each node is allowed to perform algebraic operations on received packets. Theorem 1 of [1] shows that the maximum multicast rate h is equal to the size of the minimum cut that separates the source and any terminal.

B. Minimum Cost Subgraph Generation

Let $z_{i,j}$ denote the rate at which coded packets are injected on arc (i,j). The rate vector $\mathbf{z}$, consisting of $z_{i,j}$, $(i,j)\in\mathcal{A}$, is called a subgraph [5]. Each arc $(i,j)\in\mathcal{A}$ is associated with a cost function $f_{i,j}$ that maps $z_{i,j}$ to a real number. We assume that $f_{i,j}$ is convex and monotonically increasing, which is the case when the cost is, e.g., latency or congestion. By Theorem 1 of [1], the minimum cost subgraph for a rate h asymptotically-achievable multicast with network coding is given by the following optimization problem [5]:

$$\begin{aligned}\text{minimize}\quad & \sum_{(i,j)\in\mathcal{A}} f_{i,j}(z_{i,j})\\ \text{subject to}\quad & z_{i,j}\geq x_{i,j}^{(t)},\forall(i,j)\in\mathcal{A},t\in\mathcal{T},\\ & \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{i,j}^{(t)}-\sum_{\{j|(j,i)\in\mathcal{A}\}} x_{i,j}^{(t)}=\sigma_i^{(t)},\forall i\in\mathcal{N},t\in\mathcal{T},\\ & c_{i,j}\geq z_{i,j}\geq 0,\forall(i,j)\in\mathcal{A},t\in\mathcal{T},\end{aligned}\tag{1}$$

where

$$\sigma_i^{(t)}=\begin{cases}h, & \text{if } i=s,\\ -h, & \text{if } i=t,\\ 0, & \text{otherwise.}\end{cases}\tag{2}$$

Decentralized algorithms for solving (1) have been proposed in [5]. The network codes designed in [2]–[4] can then be applied to code over the resulting subgraph.

In [1]–[4], the capacity of any arc is assumed to be unity or more generally, an integer. However, as shown in the next subsection, the minimum cost subgraph generated by solving (1) cannot always be integral. As remarked in [3], fractional capacities can be well approximated by choosing the time unit large enough. For example, if we choose the time unit n times the current value, the rate of coded packets on arc (i,j) becomes $nz_{i,j}$ and the multicast rate becomes nh. For sufficiently large n we can assume that all $nz_{i,j}$, $(i,j)\in\mathcal{A}$ are integral.

However, this standard interpretation of fractional network coding, which essentially scales the capacity of each link, is not ideal in practice. With random network coding in [4], the encoding complexity at each node is n^2 times (or n times per input packet) that without scaling. At the terminals, to decode the nh packets, a terminal needs to invert a $nh\times nh$ dense matrix, which requires $O(n^3h^3)$ operations (or $O(n^2h^2)$ operations per input packet) by using Gaussian elimination. Once the matrix inverse is obtained, applying the inverse to the received coded packets to recover the input packets requires $O(n^2h^2)$ operations (or $O(nh)$ operations per input packet). When the packet length is very long, the latter cost dominates the total decoding complexity. The decoding complexity at each terminal is n^2 times (or n times per input packet) that before scaling. The increase in encoding and decoding complexity can make the computational resources required for random network coding prohibitive. Besides, decoding is possible only after receiving at least nh coded packets, which combined with decoding delay may introduce a large latency at the terminals.

C. Integer Programming Formulation

As an alternative to scaling the solution of the LP in (1), we formulate the subgraph optimization problem as an integer program given by

$$\begin{aligned}\text{minimize}\quad & \sum_{(i,j)\in\mathcal{A}} a_{i,j}z_{i,j}\\ \text{subject to}\quad & z_{i,j}\geq x_{i,j}^{(t)},\forall(i,j)\in\mathcal{A},t\in\mathcal{T},\\ & \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{i,j}^{(t)}-\sum_{\{j|(j,i)\in\mathcal{A}\}} x_{i,j}^{(t)}=\sigma_i^{(t)},\forall i\in\mathcal{N},t\in\mathcal{T},\\ & c_{i,j}\geq z_{i,j}\geq 0,\forall(i,j)\in\mathcal{A},t\in\mathcal{T},\\ & z_{i,j}\in\mathbb{Z},\forall(i,j)\in\mathcal{A},\end{aligned}\tag{3}$$

where $\sigma_i^{(t)}$ is defined in (2), $a_{i,j}$ denotes the cost per unit rate for each arc $(i,j)\in\mathcal{A}$, and $\mathbb{Z}$ denotes the integer set.

We note that the IP with convex cost can be transformed into an equivalent IP with linear cost in an expanded graph $\mathcal{G}'=(\mathcal{N}',\mathcal{A}')$. In this transformation, for each arc $(i,j)\in\mathcal{A}$, we introduce $c_{i,j}$ arcs in $\mathcal{G}'$, each of which has unit capacity. The costs of these arcs are: $f_{i,j}(1)$, $f_{i,j}(2)-f_{i,j}(1)$, ..., $f_{i,j}(c_{i,j})-f_{i,j}(c_{i,j}-1)$. Since $f_{i,j}$ is convex and monotonically increasing, similar to [11], it can be readily shown that solving the IP with convex cost is equivalent to solving (3) in $\mathcal{G}'$. Note that the number of arcs in $\mathcal{G}'$ increases to $\sum_{(i,j)\in\mathcal{A}}c_{i,j}$. Despite the increase in $|\mathcal{A}|$, it is possible to approximately solve the relaxation of (3) in $\mathcal{G}'$ in time polynomial in $|\mathcal{N}|$ and $|\mathcal{A}|$ [11]. We can use the algorithm in [11] to solve the minimum cost flow problem in Section III-A so that the algorithm in Section III-A runs in time polynomial in $|\mathcal{N}|$ and $|\mathcal{A}|$.

In the rest of this paper, we focus on solving (3). In the rest of this paper, we focus on solving (3). Since problem (3) contains the directed Steiner-tree problem as a special case (which in turn contains the set cover problem as a special case), it is NP-hard and no polynomial time algorithm can achieve an approximation better than $O(\log k)$ unless P=NP. Thus we have the following lemma:

Lemma 1: Optimization problem (1) cannot always have an integral solution.

Proof: We show this by contradiction. It can be readily verified that (1) is a convex optimization problem, which has a polynomial-time solution. If (1) always returns an integral solution, (3) is equivalent to (1), which contradicts the NP-hardness of (3) proved in Theorem 1. □

From Theorem 1, (3) can only be solved approximately. On the other hand, achieving the optimal solution of (1) may incur higher complexity and delay. It seems that there exists a tradeoff between network cost and complexity and delay in network coding.

III. Approximation Algorithms

A. Greedy Algorithm

Let $\mathbf{z}$ denote the solution given by the greedy algorithm. Initially, $\mathbf{z}$ is an all zero vector. We first solve a minimum cost flow problem with rate h from the source s to each terminal t_i, i=1,...,k, where the corresponding flow solution for each terminal t_i is denoted as $\mathbf{x}^{(t_i)}$. Let $t^{(1)}$ denote the terminal with the minimum flow cost among all the k terminals. We update $\mathbf{z}$ by $\mathbf{z}=\max\left\{\mathbf{z},\mathbf{x}^{(t^{(1)})}\right\}$, where the max is performed componentwise, i.e. $z_{i,j}=\max\left\{z_{i,j},x_{i,j}^{(t^{(1)})}\right\}$. We then transform $\mathcal{G}=(\mathcal{N},\mathcal{A})$ into $\mathcal{G}'=(\mathcal{N},\mathcal{A}')$. In this transformation, for each arc $(i,j)\in\mathcal{A}$, we introduce arcs in $\mathcal{G}'$ according to the following rules:

- If $z_{i,j}$=0, we introduce an arc (i,j) in $\mathcal{G}'$ with the same capacity and cost as that in $\mathcal{G}$.
- If $z_{i,j}=c_{i,j}$, we introduce an arc (i,j) in $\mathcal{G}'$ with capacity $c_{i,j}$ and zero cost.
- If $c_{i,j}>z_{i,j}>0$, we introduce two arcs $(i^{(1)},j^{(1)})$ and $(i^{(2)},j^{(2)})$ in $\mathcal{G}'$. $(i^{(1)},j^{(1)})$ has capacity $z_{i,j}$ and zero cost. $(i^{(2)},j^{(2)})$ has capacity $c_{i,j}-z_{i,j}$ and cost $a_{i,j}$.

Next, we solve a minimum cost flow problem with rate h on $\mathcal{G}'$ from the source s to each remaining terminal $t_i\in\mathcal{T},t_i\neq t^{(1)}$, where the corresponding flow solution for each terminal t_i is denoted as $\mathbf{x}'^{(t_i)}$. Let $t^{(2)}$ denote the terminal with the minimum flow cost among all the $k-1$ terminals. We construct the flow solution on $\mathcal{G}$ for terminal $t^{(2)}$. For each arc $(i,j)\in\mathcal{A}$, if there exist $(i^{(1)},j^{(1)})\in\mathcal{A}'$ and $(i^{(2)},j^{(2)})\in\mathcal{A}'$, $x_{i,j}^{(t^{(2)})}=x'^{(t^{(2)})}_{i^{(1)},j^{(1)}}+x'^{(t^{(2)})}_{i^{(2)},j^{(2)}}$. $\mathbf{z}$ is also updated by $\mathbf{z}=\max\left\{\mathbf{z},\mathbf{x}^{(t^{(2)})}\right\}$. This process continues until all the terminals are included. In the nth step, for each arc $(i,j)\in\mathcal{A}$, an amount $z_{i,j}$ of the capacity $c_{i,j}$ has already been used by $t^{(1)},...,t^{(n)}$. This used portion of the capacity can be used by the remaining terminals for free. This is the intuition behind the transformation from $\mathcal{G}$ to $\mathcal{G}'$. By induction, it can be readily verified that the greedy solution returned by this algorithm is integral, and that it also satisfies all the constraints in (3). Therefore, it is feasible. The performance guarantee of this algorithm is given in Theorem 2 without proof for brevity.

Theorem 2: The greedy algorithm has an approximation ratio $O(k)$.

Proof: Let C_{grd} denote the cost of solution given by the greedy algorithm and C_{opt} denote the minimum cost of (3). Since solution returned by greedy algorithm is feasible, we have $C_{grd}\geq C_{opt}$. At the n-th step, let $C'^{(t^{(n)})}$ denote the flow cost of the terminal $t^{(n)}$ on $\mathcal{G}'$. The optimal solution of (3) also contains a feasible solution to the minimum cost flow problem from s to $t^{(n)}$ on $\mathcal{G}'$. We denote its cost as $C'^{(t^{(n)})}_{opt}$ and denote its cost on $\mathcal{G}$ as $C^{(t^{(n)})}_{opt}$. It is clear that $C'^{(t^{(n)})}_{opt}\geq C'^{(t^{(n)})}$ and $C^{(t^{(n)})}_{opt}\geq C'^{(t^{(n)})}_{opt}$. We can easily obtain the relationship $C_{opt}\geq C^{(t^{(n)})}_{opt}\geq C'^{(t^{(n)})}_{opt}$. On the other hand, it can be readily verified that $C_{grd}=\sum_{n=1}^{k}C'^{(t^{(n)})}_{opt}$. Therefore, we have $kC_{opt}\geq C_{grd}\geq C_{opt}$, and the theorem follows. □

Note that we can construct a bad example to actually attain the approximation ratio. Consider a three-layer graph as in the proof of Theorem 1. The first layer contains the source node s, the second layer contains $k+1$ nodes $m_0,m_1,...,m_k$, and the third layer contains k terminal nodes $t_1,...,t_k$. There is an arc from s to every m_i, i=0,...,k. The cost of arc (s,m_0) is $C+\epsilon$ (ϵ is a small number), and the cost of arc (s,m_i), i=1,...,k, is C. There is an arc from m_0 to every t_i, i=1,...,k, each of zero cost. There also exist arcs (m_i,t_i), i=1,...,k, each of zero cost. We want to find a minimum cost rate h=1 subgraph. Clearly, the optimal solution passes through m_0 and has a cost $C+\epsilon$. Both greedy algorithms 1 and 2 do not use the path through m_0, and result in a cost kC. The greedy algorithm has an approximation ratio k in this example. Thus, Theorem 2 is tight.

An disadvantage of the greedy algorithm is that it needs to compute minimum cost flow $\frac{k(k+1)}{2}$ times. Another observation from experimental study is that picking the minimum cost terminal at each step may not lead to a good solution. The greedy algorithm can be modified by randomly picking a terminal at each step instead of choosing the minimum cost one. This modification requires computing only one minimum cost flow at each step. It can be easily obtained that the modification runs in $O(2|\mathcal{T}||\mathcal{A}|\log|\mathcal{N}|(2|\mathcal{A}|+|\mathcal{N}|\log|\mathcal{N}|))$ time, and it also achieves an approximation ratio $O(k)$. In Section IV, experimental results show that this modification outperforms the original one.

This algorithm and its modification can also be executed in several rounds. The above described algorithm is applied in the first round. In the n-th round, the same algorithm is applied except that it is based on the solution in the $(n-1)$-th round. Let $\mathbf{x}^{(t)}[n-1]$ denote the solution for terminal t in the $(n-1)$-th round. When the algorithm comes to the terminal t_i, we define $\mathbf{z}=\max_{t\in\mathcal{T},t\neq t_i}\mathbf{x}^{(t)}[n-1]$. The same process is performed to solve $\mathbf{x}^{(t_i)}$. We update $\mathbf{x}^{(t)}[n-1]$ to the new $\mathbf{x}^{(t_i)}$. At the end of the n-th round, we set $\mathbf{x}^{(t)}[n]=\mathbf{x}^{(t)}[n-1]$. We can repeat the process K times and choose the solution with the minimum cost in all rounds.

When h=1, it is not difficult to show that the greedy algorithm reduces to the algorithm in [9] with $\kappa=|\mathcal{T}|+1$, where κ is a parameter defined in [9]. It also generalizes [7] for solving Steiner tree problem on undirected graphs, and Prim's algorithm for finding minimum spanning tree. An approximation ratio depending on the asymmetry of the graph

is also given in [9]. However, it is not hard to show that this ratio is not valid any more when $h>1$ since the proof technique used in [9] does not extend to the case with $h>1$.

B. LP Rounding

Intuitively, we can round the solution of (1) or the relaxation of (3) to an integral solution. For example, given the solution $z_{i,j}^{\mathrm{LP}}$ to the LP relaxation of (3), we can always round $z_{i,j}^{\mathrm{LP}}$ to $\lceil z_{i,j}^{\mathrm{LP}} \rceil$, where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. However, this does not always give a good approximation.

In our LP rounding algorithm, we first solve the LP relaxation of (3) and obtain the solution $z_{i,j}^{\mathrm{LP}}$. We then transform $\mathcal{G}=(\mathcal{N},\mathcal{A})$ to $\mathcal{G}'=(\mathcal{N},\mathcal{A}')$. In this transformation, for each arc $(i,j)\in\mathcal{A}$, we apply the following rules:

- If $z_{i,j}^{\mathrm{LP}}=0$, we introduce an arc (i,j) in $\mathcal{G}'$ with the same capacity and cost as that in $\mathcal{G}$;
- If $z_{i,j}^{\mathrm{LP}}=c_{i,j}$, we introduce an arc (i,j) in $\mathcal{G}'$ with capacity $c_{i,j}$ and zero cost;
- If $c_{i,j}>z_{i,j}^{\mathrm{LP}}>0$, we introduce three arcs $(i^{(1)},j^{(1)})$, $(i^{(2)},j^{(2)})$, and $(i^{(3)},j^{(3)})$ in $\mathcal{G}'$. $(i^{(1)},j^{(1)})$ has capacity $\lfloor z_{i,j} \rfloor$ and zero cost, $(i^{(2)},j^{(2)})$ has unity capacity and cost $a_{i,j}(\lceil z_{i,j}^{\mathrm{LP}} \rceil - z_{i,j}^{\mathrm{LP}})$, and $(i^{(3)},j^{(3)})$ has capacity $c_{i,j} - \lceil z_{i,j}^{\mathrm{LP}} \rceil$ and cost $a_{i,j}$,

where $\lfloor x \rfloor$ denotes the largest integer smaller than or equal to x. The greedy algorithm in Section III-A is applied on $\mathcal{G}'$.

Let C_{opt}^{LP} denote the optimal value of the LP relaxation of (3), C_{opt} denote the optimal value of (3) on $\mathcal{G}$, and $C_{opt}^{\prime,\mathrm{LPR}}$ denote the optimal value of (3) on $\mathcal{G}'$, where its value on $\mathcal{G}$ is C_{opt}^{LPR}. It is clear that $C_{opt}\geq C_{opt}^{\mathrm{LP}}$ and $C_{opt}\geq C_{opt}^{\prime,\mathrm{LPR}}$. On the other hand, by the construction of $\mathcal{G}'$, we get $C_{opt}^{\mathrm{LPR}}\leq C_{opt}^{\mathrm{LP}}+C_{opt}^{\prime,\mathrm{LPR}}$. Therefore, we have $C_{opt}^{\mathrm{LPR}}\leq 2C_{opt}$. When the greedy algorithm in Section III-A has an approximation factor ρ on $\mathcal{G}'$, we have the following theorem.

Theorem 3: The LP rounding with a ρ approximation algorithm for (3) has an approximation ratio $O(\rho)$.

A modification of the LP rounding is to change the third rule above to "If $c_{i,j}>z_{i,j}^{\mathrm{LP}}>0$, we introduce two arcs $(i^{(1)},j^{(1)})$ and $(i^{(2)},j^{(2)})$ in $\mathcal{G}'$. $(i^{(1)},j^{(1)})$ has capacity $\lfloor z_{i,j}^{\mathrm{LP}} \rfloor$ and zero cost and $(i^{(2)},j^{(2)})$ has unity capacity and cost $a_{i,j}(\lceil z_{i,j}^{\mathrm{LP}} \rceil - z_{i,j}^{\mathrm{LP}})$." Any solution to this modification at most increases the cost of C_{opt}^{LP} by $\sum_{(i,j)\in\mathcal{A}} a_{i,j}$.

The advantage of applying the greedy algorithm in Section III-A on $\mathcal{G}'$ over applying them on $\mathcal{G}$ directly is that the LP solution may guide these algorithms and remove their "myopia". The LP rounding can also be easily decentralized. To reduce the complexity of the LP rounding, the greedy algorithm in Section III-A can start from the LP solution and its dual solution. Note that our LP rounding is different from that in [8]. First, we do not round the solution in steps instead we round all $z_{i,j}$ simultaneously. Second, the algorithm in [8] is applicable for undirected graph only. Third, it is not clear how the algorithm in [8] can be decentralized.

Another feature of our LP rounding is that it can be easily combined with capacity scaling. As remarked in [3], fractional capacities can be well approximated by choosing the time unit large enough. For example, if we choose the time unit n times the current value, the rate of coded packets on arc (i,j) becomes $nz_{i,j}^{\mathrm{LP}}$ and the multicast rate becomes nh. $nz_{i,j}^{\mathrm{LP}}$ is still the optimal solution of the LP relaxation of (3) with rate nh. By using the modified LP rounding, the cost per input packet is increased by at most $\sum_{(i,j)\in\mathcal{A}} a_{i,j}/nh$ compared to the optimal value of the LP relaxation of (3). When $n\rightarrow+\infty$, the increased cost per input packet by using LP rounding is negligible. It is clear that the LP rounding with capacity scaling gives a tradeoff between encoding and decoding complexity, delay and approximation ratio.

TABLE I

PERFORMANCE OF DIFFERENT APPROXIMATION ALGORITHMS WHEN THE LP RELAXATION OF (3) RETURNS AN INTEGRAL SOLUTION.

	Random Directed Graphs			Random Geometric Graphs		
	GRD	GRD_R	LR	GRD	GRD_R	LR
Mean	1.0927	1.0770	1.0000	1.0361	1.0313	1.0000
std	0.0877	0.0762	0.0000	0.0344	0.0309	0.0000
Max	1.8655	1.6435	1.0000	1.3751	1.2359	1.0000

IV. EXPERIMENTAL RESULTS

In our test, we choose $h=5$. The capacity of each arc is chosen uniformly from the integer set $\{1,2,3,4,5\}$. We only consider linear cost, where $a_{i,j}$ on each arc is generated according to a uniform distribution on the interval $[0,1]$. The source node and the terminals are chosen randomly and uniformly from the node set. The performance is evaluated by the ratio between the cost of our approximation algorithms and the optimal value of the LP relaxation of (3), which is a lower bound on the optimal value of (3). We denote the two algorithms as GRD and LR. GRD with random terminal selection is denoted as GRD_R. In LR, we use GRD_R when the LP solution is not integral.

We first test our algorithms on both random directed graphs of Erdös and Rényi's type [14] and random geometric graphs [15] with $|\mathcal{N}|=10$, $|\mathcal{T}|=4$. In random directed graphs, we assume that there is an arc from node i to node j with probability 0.5. Random geometric graphs are always modeled as undirected graphs [15]. However, in wireless ad-hoc networks, different nodes have different transmit powers. It may not be appropriate to model a wireless ad-hoc networks as an undirected graph. We propose a class of random geometric directed graphs. We first generated $|\mathcal{N}|$ nodes and scatter them randomly over a unit square. There is an arc from node i to node j if their Euclidean distance is less than ρ_i, where ρ_i is generated according to a uniform distribution on the interval $[0,1]$, and it represents the random coverage of node i due to its random power.

TABLE II

PERFORMANCE OF DIFFERENT APPROXIMATION ALGORITHMS WHEN THE LP RELAXATION OF (3) DOES NOT RETURN AN INTEGRAL SOLUTION.

	Random Directed Graphs			Random Geometric Graphs		
	GRD	GRD_R	LR	GRD	GRD_R	LR
Mean	1.1072	1.1032	1.0180	1.0398	1.0358	1.0080
std	0.0846	0.0699	0.0181	0.0324	0.0308	0.0078
Max	1.3024	1.2767	1.0753	1.1837	1.1395	1.0428

The computational results are summarized in Tables I II, where 2000 feasible instances are generated and averaged. Table I compares the performance of different algorithms when the LP relaxation of (3) returns an integral solution, while Table II compares the performance of different algorithms when the LP relaxation of (3) does not return an integral solution.

The average running time of different approximation algorithms for random directed graphs with $|\mathcal{N}|=10,|\mathcal{T}|=4$ and $|\mathcal{N}|=20,|\mathcal{T}|=8$ is given in Table III. The running time is in seconds. Our tests are performed on a PC with a Pentium-4 CPU at 3.4 GHz.

TABLE III

AVERAGE RUNNING TIME OF DIFFERENT APPROXIMATION ALGORITHMS FOR RANDOM DIRECTED GRAPHS.

	GRD	GRD_R	LR
$\|\mathcal{N}\|=10,\|\mathcal{T}\|=4$	0.2653	0.1085	0.2061
$\|\mathcal{N}\|=20,\|\mathcal{T}\|=8$	3.1318	0.7121	2.5560

The best results are obtained by finding the LP relaxation solution of (3), and if it is not integral, using it to run the LR algorithm. We find that the LP relaxation of (3) returns the integral solution in 0.945 of the random directed graph instances and 0.9585 of the random geometric graph instances. This is not unexpected since, as stated in [14], the giant connected component of the Erdös and Rényi's random graphs is a tree with high probability. The LP relaxation of (3) always has an integer solution on a tree since there is a unique path from the source to each terminal.

The least complex algorithm is GRD_R, which surprisingly performs better than the more complex GRD. The worst case performance of our algorithms is within twice the optimal cost of the LP relaxation. This shows that both GRD and LR perform much better than the performance guarantee in Theorems 2 and 3, which is 4 in this case.From Table III, for $|\mathcal{N}|=10,|\mathcal{T}|=4$, LR has a running time twice as long as GRD_R, while for $|\mathcal{N}|=20,|\mathcal{T}|=8$, it has a running time four times as long. Our experimental results suggest that GRD_R is a favorable candidate for practical application due to its simplicity and good performance.

V. CONCLUSION

We considered the problem of minimum cost network coding with integer flows in this paper. We showed that in the LP min-cost network coding formulation, the solution is not always integral. The standard approach of choosing the time unit large enough so that fractional rates can be well approximated comes at the expense of increasing the encoding and decoding complexity, and latency at the terminals. We formulated the problem of integer network coding as an IP. Two approximation algorithms are proposed, both of which are easily decentralized. Simulation results showed that our algorithms offer good approximation ratios on random graphs. We expect that our proposed algorithms can also be useful for subgraph generation when source rates are not fixed in advance, such as in the case of rate control for elastic sources.

REFERENCES

[1] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[2] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, no. 2, pp. 371 – 381, Feb. 2003.

[3] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[4] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[5] D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2608–2623, June 2006.

[6] F. Hwang, D. Richards, and P. Winter, *The Steiner Tree Problem*. North-Holland, 1992.

[7] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," *Math. Japonica*, vol. 24, no. 6, pp. 573–577, 1980.

[8] K. Jain, "Factor 2 approximation algorithm for the generalized Steiner network problem," in *Proc. of IEEE Symposium on Foundations of Computer Science*, Nov. 1998, pp. 448–457.

[9] S. Ramanathan, "Multicast tree generation in networks with asymmetric links," *IEEE/ACM Trans. Networking*, vol. 4, no. 4, pp. 558 – 568, Aug. 1996.

[10] R. Wong, "A dual ascent approach for Steiner tree problems on a directed graph," *Math. Program.*, vol. 28, pp. 271–287, 1984.

[11] R. K. Ahuja, D. S. Hochbaum, and J. Orlin, "Solving the convex cost integer dual network flow problem," *Management Science*, vol. 49, no. 7, pp. 950–964, July 2003.

[12] V. V. Vazirani, *Approximation Algorithms*. Springer, 2004.

[13] U. Feige, "A threshold of ln*n* for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, July 1998.

[14] B. Bollobas, *Random Graphs*, 2nd ed. 2001.

[15] M. Penrose, *Random Geometric Graphs*. 2003.