On Minimizing Network Coding Resources: An Evolutionary Approach

Minkyu Kim, Chang Wook Ahn, Muriel Médard, and Michelle Effros

***Abstract*— We consider the problem of minimizing the amount of resources used for network coding while achieving the desired throughput in a multicast scenario, whose NP-hardness leads us to seeking for a method for quickly finding sufficiently good solutions. To this end, we take an evolutionary approach based on a Genetic Algorithm that works in an algebraic framework combined with randomized polynomial identity testing methods. We demonstrate the advantage of the proposed method over other existing minimal approaches by carrying out simulations on a number of different sets of network topologies. We also show, as the more important benefit of the proposed approach, its applicability to a variety of generalized optimization scenarios.**

I. Introduction

It is well known that in network B (Fig. 1(a)), by allowing network coding in contrast with simple forwarding or replicating, a multicast of rate 2 is possible, in which one can also observe that network coding is not needed at all nodes; only node z needs to combine its two inputs while all other nodes perform routing only. This observation naturally leads us to the following question: To achieve the desired throughput, at which nodes does network coding need to be done?

If network coding is handled at the application layer, by identifying the nodes where the access up to the application layer is not necessary, we can minimize the performance penalty incurred by network coding. If, on the other hand, network coding is done by a special lower layer device such as a router with the capability of mixing its inputs, it is of natural interest to reduce the number of such devices deployed while satisfying the communication demand.

Unfortunately, the problem of determining the set of minimum number of nodes where coding needs to be done is NP-hard since its decision problem, i.e., the problem to decide whether the given multicast rate is achievable without coding, reduces to a multiple Steiner subgraph problem, which is NP-hard [1].

While mostly in the network coding literature coding is assumed to be done at all possible places, the problem of reducing the amount of resources engaged in network coding has been addressed in a few recent works as follows.

M. Kim and C. W. Ahn are with Communication Laboratory, Samsung Advanced Institute of Technology, Yongin, Gyeonggi 446-712, Korea (minkyu@mit.edu, cwan.ahn@samsung.com).

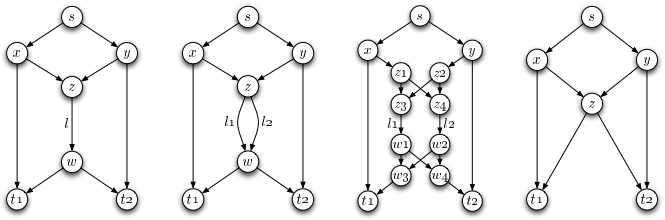
M. Médard is with the Laboratory of Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA (medard@mit.edu).

M. Effros is with Data Compression Laboratory, California Institute of Technology, Pasadena, CA 91125, USA (effros@caltech.edu).

Fragouli *et al.* [2] show that coding is required at most $d-1$ nodes in acyclic networks with 2 unit-rate sources and d sinks, which, however, is not easy to generalize to the case of more than 2 sources. They also present an algorithm to construct a minimal subtree graph, which sequentially examines each link in an arbitrary order and removes the links whose removal does not affect the achievable rate.

Langberg *et al.* [3] derive an upperbound on the number of required coding nodes for both acyclic and cyclic networks, which depends only on the desired rate and the number of sinks. They first transform the given network in such a way that each node has degree at most 3, and then obtain a minimal subgraph similarly as above, based on which the bounds are calculated. It is also shown that even approximating the minimum number of coding nodes within any multiplicative factor or within an additive factor of $|V|^{1-\epsilon}$ is NP-hard.

Both approaches, after removing links in a greedy fashion, assume network coding at all nodes with multiple incoming links in the remaining subgraph. An illustrative example below shows how these approaches may lead to a suboptimal solution in a very simple network.



(a) Network B (b) Network B' (c) Network B'' (d) Network C

Fig. 1. Sample Networks for Example 1

Example 1: Suppose that link l in network B has capacity 2, which we represent by two parallel unit-capacity links as in network B' (Fig. 1(b)). Note that the additional capacity allows for a multicast of rate 2 without network coding. Fragouli *et al.*'s approach, however, always removes the first visited link out of l_1 and l_2, which renders coding at z necessary in the remaining graph. On the other hand, Langberg *et al.*'s method first decomposes nodes z and w as in Fig. 1(c), in which there are many sequences of link removal that result in a subgraph where coding is required: e.g., if l_1 is the first visited link, node z_4 must perform coding. Empirical tests show that, via this approach with a randomly chosen order of link traversal, coding is found to be required with probability about 0.68.

Let us consider another network C (Fig. 1(d)), where further

link removal is not possible, but coding at the merging node z is not needed. We can observe that obtaining a subgraph with minimal, or even minimum, link usage does not rule out the nodes where coding is possible but not necessary. □

Bhattad *et al.* [4], on the other hand, give linear programming formulations for the problems of optimizing over various resources used for network coding, based on the model allowing continuous flows. Their optimal formulations, however, involve an exponential number of variables and constraints growing with the number of sinks, which makes it hard to apply the formulations to the case of a large number of sinks, even at the price of sacrificed optimality.

Rather than tackling an NP-hard problem toward optimality, we focus on quickly finding a sufficiently good solution. One can observe in the above example that finding a good order of link traversal, out of exponentially many possible sequences in general, is critical to the quality of the solution by Langberg *et al.*'s approach. Likewise, the problem of deciding where to perform coding involves the selection out of a large number of choices. In this paper, we approach to this problem by employing a method that manages a set of candidate solutions of only a suitably small size being subject to sequential enhancement in an evolutionary manner.

The above example also illustrates a possible tradeoff between network coding and link usage; i.e., if we first try to reduce link usage as in Fragouli *et al.*'s method, coding may become necessary in the remaining subgraph, while minimizing the number of coding nodes first will result in more link usage. An optimal choice would certainly depend on the relative cost of each resource, and as will be discussed later, our proposed method can be well generalized to the case where optimization over both kinds of costs is needed.

This paper is organized as follows: Section II presents the formulation of the problem with a brief introduction to Genetic Algorithms, Section III describes the details of the approach we propose in this paper, whose performance Section IV demonstrates in comparison with other minimal approaches, Section V generalizes our method to various optimization scenarios, and Section VI concludes with a summary of the results and a discussion of further work.

II. Problem Formulation

Throughout the paper, we assume that a network is given by an acyclic directed multigraph $G = (N, A)$ where each link has a unit capacity, and to represent links with larger capacities, multiple links are allowed between a pair of nodes. At each link, only integer amount of flow is allowed, hence there is either no or unit rate of flow. We consider the single multicast scenario in which a single source $s \in N$ wishes to transmit data at rate R to a set $T \subset N$ of sink nodes, where $|T| = d$. If there exists a transmission scheme, either involving network coding or not, that enables all d sinks to receive all the information sent, rate R is said to be achievable.

We wish to determine the set of minimum number of nodes where coding is required in order to achieve the given rate R, which we assume is achievable if coding is allowed at all nodes. Note that, with network coding at all possible nodes, the maximum achievable multicast rate is the minimum of the individual max-flow bounds between the source and each of the sinks [5], and for the decision version of the problem – the problem to verify the achievability of the given multicast rate – an algebraic formulation is given in [6]. We will consider how this algebraic formulation can be applied to the case where network coding is done only at some subset of the nodes.

We only consider linear coding, which is sufficient for multicast [7], where a node's output to an outgoing link is represented as a linear combination of the inputs from its incoming links. Nodes with a single incoming link have no other input to be combined with the incoming information, and hence it is clear that no coding is required for such nodes (a formal proof can be found in [8]).

At nodes with multiple incoming links, if the output of an outgoing link happens to be a linear combination of the inputs such that all but one of the coefficients are zero, effectively no coding operation occurs for that link; even if the only nonzero coefficient is not identity, there is another coding scheme with the coefficient replaced by identity [3]. Hence, to find the nodes where coding is not necessary, we need to verify, only at each of the nodes with multiple incoming links, if we can restrict its outputs to depend on a single input without destroying the achievability of the given rate.

More specifically, the above verification is done as follows: We first construct, as demonstrated in [6], the labeled line graph $G' = (N', A')$ corresponding to G. Then, to each link in G' we assign a link coefficient, denoted by ξ_i, and construct a system matrix for each of d connections, which is R-by-R matrix describing the relationship between the input from the source and the output to that sink. If we denote by $P(\underline{\xi})$ the product of the determinants of those d matrices, the given multicast rate is achievable if and only if $P(\underline{\xi})$ is nonzero over the ring of polynomials in variables $\underline{\xi}$ [6].

Note that, for a node in G with multiple incoming links, each of its outgoing links is represented by a node in G' with multiple incoming links. Therefore, we need to inspect only the nodes in G' with multiple incoming links; to be specific, for node v in G with incoming links $l_1, ..., l_n$, at each of the n nodes in G' corresponding to those n links, we examine the coefficients assigned to its incoming links, out of which if there exists a single coefficient except which zeroing out all other coefficients does not render $P(\underline{\xi})$ to be a zero polynomial, we can conclude that coding is not required at node v assuming that all other nodes perform coding.

The difficulty arises when several nodes are considered together; whether coding is needed at a node depends on whether coding is done at other nodes and thus the above verification procedure cannot be applied separately to each node. For example, in network D (Fig. 2) with three sinks and the target multicast rate 2, both nodes a and b are identified to be non-coding nodes when verified separately, but not together. As the number of involved nodes increases, checking the necessity of coding may require the evaluation

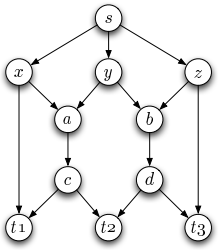


Fig. 2. Network D showing correlation between coding at different nodes.

of exponentially many selections of link coefficients.

Note that the only aspect of the coefficients we are interested is whether their being zero or not, hence if there are m coefficients associated with the nodes with multiple incoming links, we are to explore an m-dimensional binary space. As we try to find the nodes where coding is necessary, we are facing 2^m choices about how to proceed, but little theoretical guidance can be given on how to make them optimally. We employ a search method, based on a Genetic Algorithm (GA), that serves to efficiently reduce the size of the space to be actually searched using an evolutionary mechanism, as details will be provided in the subsequent sections.

A. A Brief Introduction to Genetic Algorithms

Genetic Algorithms (GAs) are stochastic search methods employing natural genetic ideas such as gene recombination, mutation and survival of the fittest. GA has been applied to a large number of scientific and engineering problems, including many combinatorial optimization problems in networks (e.g., [9], [10]).

GAs [11], [12] operate on a set of candidate solutions, called *population*, where each solution is represented typically by a bit string, called *chromosome*. Each chromosome is assigned a *fitness value* that measures how well the chromosome solves the problem at hand, compared with other chromosomes in the population. From the current population, a new population is generated using typically three genetic operators: *selection*, *crossover* and *mutation*. Chromosomes are selected randomly (with replacement) for the new population in such a way that fitter chromosomes have more chances to be selected. For crossover, survived chromosomes are randomly paired, and in each pair two chromosomes exchange a subset of the bit strings to create two offspring. Chromosomes are then subject to mutation which refers to random flips of the bits applied individually to each of the new chromosomes. The process of evaluation, selection, crossov! er and mutation forms one *generation* in the execution of a GA. The above process is iterated with the newly generated population successively replacing the current one, and terminates when a certain stopping criterion is reached, e.g., after a predefined number of generations.

There are several aspects of our problem suggesting that a GA-based method be a promising candidate: GA has proved to apply well if the space to be searched is large, but known not to be perfectly smooth or unimodal, or not well understood, and if finding a global optimum is not critical [11]. Note that the search space consisting of m-dimensional binary vectors is not smooth or unimodal with respect to the number of coding nodes and the structure of the space consisting of the feasible vectors is not well understood. Also, the NP-hardness of the problem allows us to only hope for quickly finding a sufficiently good solution, not necessarily optimal.

Note also that, while it is hard to characterize the structure of the search space, once provided with a solution we can rather easily verify its feasibility and count the number of the coding nodes therein. Thus, if the use of genetic operations can suitably limit the size of the search space, a solution can be obtained fairly efficiently.

III. Proposed Approach

To begin with, recall that our decision on the necessity of coding at a node is based on the inspection of all of its outgoing links, which indicates that links rather than nodes can be more correctly referred to as the location for coding. It is also pointed out in [3] that the number of coding links is a more accurate estimator of the amount of computation incurred by coding. In this section, we set our objective to minimizing the number of coding links, which can easily generalize to the case of coding nodes, as will be shown in the next section.

We employ the structure of the standard GA introduced by Holland [13] (see Fig. 3) with its elements specifically designed to fit our problem, as will be described below. Note that GA's performance depends on the details of its elements such as selection mechanism, crossover operator, and numerical parameters, etc. Theoretical works, however, were not yet able to provide a useful prediction about which combination of such elements would be best suited to a specific problem [11]. We thus tested several choices of the elements reported to work well in many other studies, and picked the one that worked best for our problem.

```
{
 initialize population;
 evaluate population;
 while termination criterion not reached
    {
      select solutions for next population;
      perform crossover and mutation;
      evaluate population;
    }
}
```

Fig. 3. Standard Genetic Algorithm Structure [12]

A. Notations and Preliminaries

We first construct the labeled line graph G', in which we refer to each node with multiple incoming links as a *coding point* and let C be the the set of all coding points. For the ith coding point $c_i \in C$, we let M_i be the set of the link coefficients associated with the incoming links to c_i, and let

M denote the union of all such M_i's where $|M| = m$. We assume that the components of the vector $\underline{\xi}$ consisting of all link coefficients are rearranged such that the first m components of $\underline{\xi}$ belong to M, i.e., $\xi_j \in M (1 \leq j \leq m)$. As discussed in Section II, each chromosome is represented by an m-dimensional binary vector, whose kth component is associated with ξ_k. Once a chromosome $\underline{y}$ is given, we refer to each coding point c_i as *inactive* if the number of 1's in the $\underline{y}$'s components associated with the set M_i is at most one, and *active* oth! erwise.

B. Initial Population

The initial population is randomly constructed such that each component of the chromosomes is assigned 0 or 1 with equal probabilities. Note, however, that the size of the population, typically not exceeding a few hundreds, is much smaller than the size of the entire space, and thus it is very unlikely that a feasible chromosome is seeded into the initial population, the lack of which may cause the algorithm to fail to yield a single feasible solution for a considerable number of early generations.

We thus insert into the randomly generated population the vector of all 1's, which is feasible by assumption but renders all coding points active, and this insertion turns out to improve the performance of the algorithm very significantly. For instance, without the all-one vector, the algorithm almost always ends with the population of only infeasible chromosomes for a mid-sized problem with $m = 80$.

C. Fitness Evaluation

We define the fitness value F of chromosome $\underline{y}$ as

$$F(\underline{y}) = \begin{cases} \text{number of active } c_i\text{'s}, & \text{if } \underline{y} \text{ is feasible,} \\ \infty, & \text{if } \underline{y} \text{ is infeasible.} \end{cases}$$

To verify the feasibility of a given chromosome $\underline{y}$, we evaluate the polynomial $P(\underline{\xi})$ such that

$$P(\underline{\xi})|_{\xi_k=0 \text{ for } k \text{ s.t. } y_k=0 (1 \leq k \leq m).}$$

Note that the determinant of the ith transfer matrix M_i ($1 \leq i \leq d$), which is defined as $M_i = A(I - F)^{-1} B_i^T$ and of size R-by-R for multicast rate R [6], may contain $R!$ terms, each of which already consists of up to $O(\mu^2 |E|)$ monomials, where μ is the maximum number of links outgoing incident to the source or incoming incident to the sinks. Hence, keeping $P(\underline{\xi})$ in a polynomial form is very inefficient (or even impossible) for its exponential size.

Rather than treating P explicitly in a polynomial form, we thus keep A, F, and B_i's in a matrix form and rely on one of the following approaches: The first method is to assign random elements from a finite field $\mathbb{F}_q$ to nonzero elements of A, F, and B_i's, and then to declare $P(\underline{\xi})$ to be nonzero with *no error* if the product of the determinants evaluates a nonzero value in $\mathbb{F}_q$, and zero otherwise with an error probability upperbounded by $1 - (1 - d/q)^{\nu}$, where ν is the maximum number of links in any set of links constituting a flow solution from all sources to any receiver [14]. Note, however, that in our context of optimization this one-sided error, if might happen, only affects the solution to be more conservative, which is far less critical than the opposite case where an infeasible solution is mistakenly admitted feasible. We can lower the error bound as much as we desire by increasing the field size ! or repeating the random test at an additional cost of computation.

Alternatively, since we are interested in the existence of a network code in a field of any size, the above randomized test, as those originally developed by Schwartz, Zippel and many others, can generalize as follows: By assigning random integers from a finite set S and operating in the real field, the randomized test, which now has the error probability bound of $(1 - d\nu/|S|)$, can run substantially faster than that performing matrix computations in a large finite field. Note, however, that very large components of M_i's due to the matrix inversion, $(I - F)^{-1}$, can prevent exact calculation of determinants numerically, and in such a case we may choose to use, for a practical purpose, a numerical method such as the condition number, whose being infinite implies singularity of the matrix. The condition number is efficiently calculated by singular value decomposition and considered a numerically reliable indicator of matrix singularity [15], [16].

In addition, there are many more recent algorithms for this purpose, called polynomial identity testing, some of which use a reduced number of random bits [17], [18] and some take deterministic approaches [19], [20]. These algorithms are not used for our numerical simulations, nevertheless the possibility of adopting any efficient testing method including those mentioned above is fully open.

D. Genetic Operators and Numerical Parameters

We employ a rank-based selection mechanism which in many cases allows for more successful search than the original fitness-proportionate selection methods [11]; in particular, an exponential ranking method is used, as suggested by [21], where the probability of a particular chromosome's selection decreases exponentially with its rank in the population. We also put *elitism* into effect by retaining the best chromosome unaltered at each generation, which is found by many researchers to significantly improve the GA's performance [11].

We use parameterized uniform crossover, where each pair of chromosomes is selected for crossover with probability 0.8 and the two chromosomes in a selected pair exchange each bit independently with probability 0.8. Parameterized uniform crossover is commonly used in recent GA applications [11] and indeed turns out to work better for our problem than other traditional crossover operators such as one- or two-point crossovers. For mutation, simple binary mutation, where each bit in each chromosome is flipped with probability 0.01, is used.

The population size is set to 150, and the iteration is terminated if no progress is made in the best value of the population for 100 generations or if the generation number reaches a limit: 300 for the simulations in the next section.

	Set I								Set II				Set III			
	3 Copies		7 Copies		15 Copies		31 Copies		LATA-X		ISP 1755		(20,12,4)		(40,12,3)	
	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
Proposed	0	0.65	0	2.15	3	5.35	12	17.20	0	0.35	0	0.25	0	1.20	0	1.05
Minimal 1	7	7.00	19	19.00	43	43.00	91	91.00	11	11.25	4	8.55	20	23.30	12	15.10
Minimal 2	0	2.15	2	4.70	7	11.60	28	52.80	0	1.10	0	0.80	0	1.85	0	1.90

TABLE I

NUMBER OF CODING LINKS CALCULATED BY THE PROPOSED METHOD AND TWO MINIMAL APPROACHES

IV. PERFORMANCE EVALUATION

We demonstrate the performance of our approach by carrying out simulations on various network topologies. For comparison, we also perform numerical tests using the two previously mentioned minimal approaches by Fragouli *et al.* [2] ("Minimal 1") and Langberg *et al.* [3] ("Minimal 2"), in both of which link removal is done in a random order. For each of the three methods, the best and the average values obtained by 20 iterations are shown in Table I.

A. Set I of Networks

Consider the network constructed by cascading a number of copies of network B' in Example 1(Fig. 1(b)) such that the source of a copy of B' is replaced by another's sink (see Fig. 4). It is clear that the networks constructed as such have the maximum multicast rate 2, which is achievable without coding at all; i.e., the optimal value is always zero. With the source being the node at the top and the sinks being all the leaf nodes, networks with 3, 7, 15, and 31 cascaded copies of B' are chosen for our simulations, in which the number of sinks is 4, 8, 16, 32, respectively.

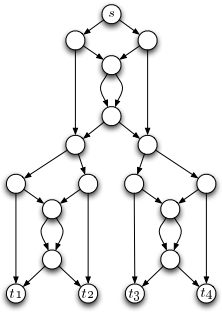


Fig. 4. Example from the set I of networks: 3 cascaded copies of B'

B. Set II of Networks

We also apply our method to sample backbone topologies: the local access transport area network X (LATA-X) and ISP 1755 (Ebone) topology obtained from the Rocketfuel Project [22]. Assuming for simplicity that each link has unit capacity, we choose the orientation of each link such that no cycle is generated while the given multicast rate is achievable between a source and the given number of sinks that are arbitrarily selected; in particular, the parameters used are (9 sinks, rate 2) for LATA-X and (4 sinks, rate 3) for ISP 1755.

C. Set III of Networks

As another set of sample networks, we employ the topologies generated by the algorithm in [23], which constructs connected acyclic directed graphs uniformly at random; two networks with parameters (20 nodes, 80 links, 12 sinks, rate 4) and (40 nodes, 120 links, 12 sinks, rate 3) are used for simulations.

We observe that our approach performs far better than Minimal 1 approach, both in terms of the best and the average values. In comparison with the second minimal approach, our approach produces considerably better solutions on average while the gap between the best values by the two methods becoming wider for larger-sized problems. We can also observe that the networks in sets II and III, as opposed to those in set I, lead to no difference in the best values obtained by 20 iterations of the proposed and Minimal 2 approaches, which may indicate that the scenario as in network B' in Example 1 is not much likely to be the case in general topologies. However, in the case where doing such many iterations is not easy, the proposed method can be much more useful. Note that the benefit of the proposed approach is not only its superior performance in reducing the number of coding links, but more importantly its applicability to various generalized scenarios, as will be di! scussed in the next section.

V. GENERALIZATION

The more important advantage of our proposed approach lies in that, as opposed to the previously mentioned minimal approaches, it can be readily applied to a variety of generalized problems, which involve non-coding links/nodes and thus are hard to solve optimally.

1) Number of Coding Nodes: The proposed method can easily generalize to the case of minimizing coding nodes, which initially was our objective. For feasible chromosome $\underline{y}$, we alternatively define $F(\underline{y})$ as the number of nodes with multiple incoming links any of whose associated coding points is active. Table II shows the number of required coding nodes computed by such modified method for the set I of networks. (For the minimal approaches, due to the special structure of the networks in set I, the number of coding nodes happens to be the same as that of coding links. Thus, see Table I for comparison.)

2) Different Coding Costs: If the cost for coding is different at each of the links, one would be interested in minimizing the total overhead incurred by coding, which can be calculated by summing up the coding cost at each of the active coding points

3 Copies		7 Copies		15 Copies		31 Copies	
Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
0	0.85	0	2.60	3	6.00	12	19.05

TABLE II

PERFORMANCE OF THE PROPOSED METHOD FOR CODING NODES

and is to be used as the fitness value of a feasible chromosome. A similar generalization works for the case of coding nodes. On the other hand, the previous minimal approaches may not have a natural generalization to this scenario unless the coding costs can be clearly ordered, in which case traversing the links/nodes in descending order of cost seems reasonable.

3) Routing Solution and Network Code: A solution produced by our method determines the link coefficients in M either to be zeroed out or to remain indeterminate, which in the latter case are to be assigned elements from a proper finite field by any existing coding method. Note that each of the link coefficients not belonging to M, which we call *routing coefficients*, also has binary choices: It is now either zero or *identity* rather than an indeterminate. Hence, by simply adding the routing coefficients to the solution vector, we can obtain a feasible routing solution that determines which links are used for routing, for now without any optimization. Furthermore, if the randomized fitness evaluation method in a finite field is used with all nonzero routing coefficients being replaced by identity, a feasible network code is obtained, without any additional code construction procedure, at the end of the iteration.

4) Consideration of Link Costs: The cost for link usage is clearly subject to optimization, which alone can be efficiently solved by assuming coding at all possible places [24] while joint optimization over the coding and link costs is difficult; e.g., the formulation in [4] entails an exponential number of variables and constraints. We note that the optimization of link cost jointly with coding cost can be incorporated into our GA-based framework by adjusting the fitness value as follows: Given a feasible chromosome that includes the routing coefficients, for each of the links if any of its associated coefficients is nonzero, we add the associated link cost to the fitness value in which the coding cost has already been taken into account.

VI. CONCLUSIONS AND FURTHER WORK

We have proposed an evolutionary approach to the problem of minimizing the amount of resources used for network coding and compared its performance with other existing minimal approaches. Our result show that the proposed approach achieves significantly superior performance over the minimal approaches. The more important benefit the proposed approach offers is its applicability to a variety of generalized optimization scenarios.

There are several topics for further research. GA components of the proposed approach, such as the method for constructing the initial population, can be further specialized for the problem at hand to improve the algorithm's performance. The framework of the proposed approach may be modified so as to work with as cyclic graphs or to allow for semi-decentralized operation with only limited amount of feedback. Also, more recent GA techniques, e.g., linkage learning GA which offers improved scalability exploiting the correlation between variables that is to be learned as the algorithm progresses, are worth to investigate for their applicability in the context of network coding.

REFERENCES

[1] M. B. Richey and R. G. Parker, "On multiple steiner subgraph problems," *Networks*, vol. 16, no. 4, pp. 423–438, 1986.

[2] C. Fragouli and E. Soljanin, "Information flow decomposition for network coding," *IEEE Trans. Inform. Theory*, to appear.

[3] M. Langberg, A. Sprintson, and J. Bruck, "The encoding complexity of network coding," in *Proc. IEEE ISIT '05*.

[4] K. Bhattad, N. Ratnakar, R. Koetter, and K. R. Narayanan, "Minimal network coding for multicast," in *Proc. IEEE ISIT '05*.

[5] 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, 2000.

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

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

[8] Y. Wu, P. A. Chou, and S. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Trans. Commun.*, to appear.

[9] R. Elbaum and M. Sidi, "Topological design of local-area networks using genetic algorithms," *IEEE/ACM Trans. Networking*, vol. 4, no. 5, pp. 766–778, 1996.

[10] B. Dengiza, F. Altiparmak, and A. E. Smith, "Efficient optimization of all-terminal reliable networks, using an evolutionary approach," *IEEE Trans. Rel.*, vol. 46, no. 1, pp. 18–26, 1997.

[11] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[12] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *IEEE Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[13] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Univ. of Michigan Press, 1975.

[14] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE ISIT '03*.

[15] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1993.

[16] J. W. Demmel, *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

[17] Z.-Z. Chen and M.-Y. Kao, "Reducing randomness via irrational numbers," *SIAM J. Comput.*, vol. 29, no. 4, pp. 1247–1256, 2000.

[18] D. Lewin and S. Vadhan, "Checking polynomial identities over any field: towards a derandomization?" in *Proc. ACM STOC '98*, pp. 438–447.

[19] R. Lipton and N. Vishnoi, "Deterministic identity testing for multivariate polynomials," in *Proc. SODA '03*, pp. 756–760.

[20] V. Kabanets and R. Impagliazzo, "Derandomizing polynomial identity tests means proving circuit lower bounds," in *Proc. ACM STOC '03*, pp. 355–364.

[21] C. R. Houck, J. A. Joines, and M. G. Kay, "A genetic algorithm for function optimization : a matlab implementation," NCSU-IE, Tech. Rep. 95-09, 1995.

[22] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proc. of ACM/SIGCOMM '02*, pp. 133–145.

[23] G. Melançon and F. Philippe, "Generating connected acyclic digraphs uniformly at random," *Inf. Process. Lett.*, vol. 90, no. 4, pp. 209–213, 2004.

[24] D. S. Lun, M. Médard, T. Ho, and R. Koetter, "Network coding with a cost criterion," MIT-LIDS, Tech. Rep. P-2584, 2004.