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 resources used for network coding while achieving the desired throughput in a multicast scenario. Since this problem is NP-hard, we seek 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 rather than only forwarding and replicating, a multicast of rate 2 is possible. In this example, 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 occur?

If network coding is handled at the application layer, we can minimize the performance penalty incurred by network coding by identifying the nodes where access up to the application layer is not necessary. 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.

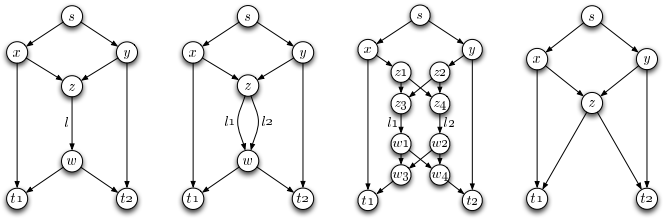
Unfortunately, the problem of determining a minimal set of nodes where coding is required is NP-hard: its decision problem, which decides whether the given multicast rate is achievable without coding, reduces to a multiple Steiner subgraph problem, which is NP-hard [1].

While most of the network coding literature assumes coding is done at all nodes, the problem of reducing the amount of resources engaged in network coding has been addressed in a few recent works.

Fragouli *et al.* [2] show that coding is required at no more than $d-1$ nodes in acyclic networks with 2 unit-rate sources and d sinks. This result, however, is not easily generalized to more than 2 sources. They also present an algorithm to construct a minimal subtree graph. For target rate R, they first select a subgraph consisting of R link-disjoint paths to each of d sinks. In the labeled line graph corresponding to the subgraph, each link is sequentially examined and removed if its 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. The bounds depend only on the desired rate and the number of sinks. They first transform the given network to give a new network in which each node has degree at most 3. Then, similarly as above, they sequentially remove the links without which the target rate is still achievable to obtain a minimal subgraph. The bounds are calculated for the resulting network. They also show 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 graph. 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.

In Fragouli *et al.*'s approach, either of links l_1 and l_2 may be removed while selecting the subgraph, which renders coding at z necessary in the remaining graph. If, on the other hand, both links l_1 and l_2 are retained in the subgraph, whether coding is required depends on the order in which the links are visited to construct a minimal subtree graph; for a randomly chosen order of link inspection, coding is required with probability $\frac{1}{2}$.

This work was done while M. Kim was doing an internship at Samsung Advanced Institute of Technology.

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

C. W. Ahn is with Communication Laboratory, Samsung Advanced Institute of Technology, Yongin, Gyeonggi 446-712, Korea (cwan.ahn@samsung.com).

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

Langberg *et al.*'s method first decomposes nodes z and w as in Fig. 1(c); for this network, there are many sequences of link removals 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 the probability that coding is required for random link removals is about 0.68.

Let us consider another network C (Fig. 1(d)). Here 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] give linear programming formulations for the problems of optimizing over various resources used for network coding, based on a model allowing continuous flows. Their optimal formulations, however, involve a number of variables and constraints that grows exponentially 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, 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 solutions by the two minimal approaches. Likewise, the problem of deciding where to perform coding involves a selection out of a large number of choices. Our method manages a set of candidate solutions of a suitably small size, sequentially enhancing these solutions in an evolutionary manner.

The above example also illustrates a possible tradeoff between network coding and link usage. Reducing usage as in the subgraph selection of Fragouli *et al.*'s method may increase coding in the remaining subgraph; minimizing coding first may increase link usage. An optimal choice depends on the relative cost of each resource; our proposed method can be generalized to the case where optimization over both kinds of costs is needed.

This paper is organized as follows. Section II presents the problem formulation with a brief introduction to Genetic Algorithms. Section III describes the details of our proposed approach. Section IV gives experimental results and comparison with other minimal approaches. Section V generalizes our method to other optimization scenarios. Section VI concludes with a summary of the results and a discussion of future work.

II. Problem Formulation

Throughout the paper, we assume that a network is given by an acyclic directed multigraph $G = (V, E)$ where each link has a unit capacity. To represent links with larger capacities, multiple links are allowed between a pair of nodes. Only integer flows are allowed, hence there is either no flow or a unit rate of flow on each link. We consider the single multicast scenario in which a single source $s \in V$ wishes to transmit data at rate R to a set $T \subset V$ of sink nodes, where $|T| = d$. Rate R is said to be achievable if there exists a transmission scheme that enables all d sinks to receive all of the information sent.

Given an achievable rate R, we wish to determine a minimal set of nodes where coding is required in order to achieve this rate. With network coding at all nodes, the maximum achievable multicast rate is the minimum of the individual max-flow bounds between the source and each of the sinks [5]. An algebraic formulation of the general network coding problem appears 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, where a node's output on an outgoing link is a linear combination of the inputs from its incoming links. Linear coding is sufficient for multicast [7]. It is clear that no coding is required at a node with only a single input since these nodes have nothing to combine with (a formal proof appears in [8]).

If the linearly coded output from a node with multiple incoming links weights all but one incoming message by zero, then effectively no coding occurs on that link; even if the only nonzero coefficient is not identity, there is another coding scheme that replaces the coefficient by identity [3]. Hence, to find the nodes where coding is not necessary, we need to verify at each of the nodes with multiple incoming links whether we can restrict the given node's outputs to depend on a single input without destroying the achievability of the given rate.

We perform the above verification as follows: We first construct the labeled line graph $G' = (V', E')$ corresponding to G [6]. 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. Each system matrix is an R-by-R matrix describing the relationship between the input from the source and the output to that sink. Let $P(\underline{\xi})$ denote 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].

Each output of any node in G with multiple incoming 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. If there exists a vector of coefficients for the given node in G' such that all but one of the coefficients is zero and the resulting $P(\underline{\xi})$ is a nonzero 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, when either node a or node b is tested separately, we find that neither must be a coding node. Looking at the network as a whole, however, we find that coding is required at at least one of a and b to achieve capacity. As the number of involved nodes increases, checking the necessity of coding may require the evaluation of exponentially many selections of link coefficients.

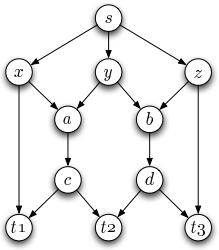


Fig. 2. Network D showing correlation between coding at different nodes.

Note that we are only interested in whether or not all but one of the coefficients can zero. Hence if there are m coefficients, we are to explore an m-dimensional binary space. As we try to find the nodes where coding is necessary, we consider 2^m choices with little theoretical guidance on the optimal choice. We employ a search method, based on a Genetic Algorithm (GA), that serves to efficiently reduce the size of the space to be searched using an evolutionary mechanism. We begin with an introduction to GAs. Details on our algorithm follow in subsequent sections.

A. A Brief Introduction to Genetic Algorithms

GAs are stochastic search methods that mimic genetic phenomena such as gene recombination, mutation and survival of the fittest. GAs have 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 a *population*. Each solution is typically represented by a bit string, called a *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 typically using three genetic operators: *selection*, *crossover* and *mutation*. Chromosomes for the new population are selected randomly (with replacement) in such a way that fitter chromosomes are selected with higher probability. For crossover, survived chromosomes are randomly paired, and then two chromosomes in each pair exchange a subset of their 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, crossover 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. The GA 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 may be a promising candidate: GA has proven to work well if the space to be searched is large, but known not to be perfectly smooth or unimodal, or if the space is 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 good solution, even if that solution is 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 easily verify its feasibility and count the number of 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

Since our decision on the necessity of coding at a node is based on the inspection of all of its outgoing links, and the number of coding links is a more accurate estimator of the amount of computation incurred by coding [3], our objective in this section is to minimize the number of coding *links*. We discuss the generalization to coding *nodes* 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 the selection mechanism, crossover operator, numerical parameters, etc. Theory to accurately predict which combination of such elements is best suited to a specific problem is not yet available [11]. We thus test several choices that work well in many other studies and pick the one that works 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 χ_i be the set of coefficients associated with the incoming links to c_i, and let χ denote the union of all such χ_i's where $|\chi| = m$. We assume that the components of vector $\underline{\xi}$, which consists of all link coefficients, are rearranged such that the first m components of $\underline{\xi}$ belong to χ, i.e., $\xi_j \in \chi (1 \leq j \leq m)$. 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 χ_i is at most one, and *active* otherwise.

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 hundred, 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. As a result, the algorithm may 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 renders all coding points active. This solution is feasible by assumption, and in our experiments this insertion improves 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

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

Note that each transfer matrix M_i ($1 \le i \le d$), which is defined as $M_i = A(I - F)^{-1}B_i^T$ in [6], has size R-by-R for multicast rate R, and each of its elements is a polynomial consisting of $O(\mu|E|^2)$ terms, where μ is the maximum number of ways to traverse from any link to another in the network, which in general grows exponentially with the size of the network. The determinant of M_i thus contains $O((\mu|E|^2)^R \cdot R!)$ terms, which makes keeping $P(\underline{\xi})$ in polynomial form very inefficient (or even impossible) for its exponential size.

Rather than treating $P(\underline{\xi})$ explicitly in polynomial form, we keep A, F, and B_i's in 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 the nonzero elements of A, F, and the B_i's, and then to declare $P(\underline{\xi})$ to be nonzero if the product of the determinants evaluates to a nonzero element in $\mathbb{F}_q$, and zero otherwise. The probability of an error in declaring $P(\underline{\xi}) \neq 0$ is 0; the probability of an error in declaring $P(\underline{\xi}) = 0$ is bounded above by $1 - (1 - d/q)^\nu$, where ν is the maximum number of links in any set of links constituting a flow solution from the source to any receiver [14]. In our optimization, this one-sided error makes our solution more conservative, which is far less critical than the opposite case, where an infeasible solution might mistakenly be declared 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 (e.g., [15]), can generalize as follows: If we assign random integers from a finite set S and operate in the real field, the randomized test, which now has the error probability no greater than $(1 - d\nu/|S|)$, 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 a numerical method such as the condition number, which signifies that the matrix is singular. The condition number is efficiently calculated by singular value decomposition and is considered a numerically reliable indicator of matrix singularity [16], [17].

In addition, there are many more recent algorithms for this purpose, called polynomial identity tests, some of which use a reduced number of random bits [18], [19] and some of which take deterministic approaches [20], [21]. 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 [22], 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. This fact may indicate that the correlation between coding points is not necessarily associated with the proximity between their corresponding components in a chromosome. For mutation, we use simple binary mutation, where each bit in each chromosome is flipped independently with probability 0.01.

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.

IV. Performance Evaluation

We demonstrate the performance of our approach by carrying out simulations on various network topologies. For

	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	3	3.00	7	7.00	15	15.00	31	31.00	0	0.90	0	1.05	0	1.35	1	1.85
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

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 Minimal 1, the subgraph is selected also by a minimal approach, which starting from the original graph sequentially removes the links whose removal does not destroy the achievability. For each of the three methods, the best and the average values obtained in 20 random trials 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 each subsequent copy of B' is replaced by an earlier copy's sink (see Fig. 4). It is clear that the networks constructed in this way have maximum multicast rate 2, which is achievable without coding; i.e., the optimal number of coding links is always zero. For simulations, we use fixed-depth binary trees containing 3, 7, 15, and 31 copies of B' and 4, 8, 16, and 32 sinks, respectively; in each network, the tree's root node is the network's source and the end nodes are the network's sinks.

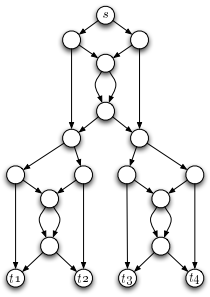


Fig. 4. Example from the set I of networks: 3 copies of B' in a depth-1 binary tree

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 [23]. 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 [24], 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.

In our experiments, the performance of our approach is everywhere at least as good and often far better than that of both Minimal 1 and Minimal 2 both in the best and in the average values. For networks in set I, note that the gap between the best values of our algorithm and the two minimal approaches grows with the size of the network. For networks in sets II and III, in most cases, there is no difference in the best values obtained by 20 trials of the proposed and the minimal approaches, which may indicate that the scenario captured by network B' in Example 1 is not very likely to occur in general topologies. The proposed method may be even more useful when running many iterations is computationally infeasible. The benefit of the proposed method goes beyond its superior performance in reducing the number of coding links. A more important benefit is its applicability to various generalized scenarios, as will be discussed in the next section.

V. GENERALIZATION

Unlike Minimal 1 and Minimal 2, our proposed approach can be readily applied to a variety of generalized problems that 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 that require coding on one or more outgoing links. Table II shows the number of required coding nodes computed by this modified method for the set I of networks. (For the minimal approaches, the number of coding nodes happens to be the same as that of coding links; i.e., at any merging node, if one outgoing link does coding, the other outgoing link is always removed. 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

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

by summing up the coding cost at each of the active coding points and using this sum 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 do 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: Our method determines if each of the link coefficients in χ is either to be zeroed out or to remain indeterminate. Note that the link coefficients not belonging to χ, which we call *routing coefficients*, also have binary choices: either zero or *identity*. 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 [25] 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. Note that, for Minimal 1, one may consider a two-phase method such that link cost is reduced while selecting the subgraph and coding cost is reduced separately by the minimal approach. Note that, in the previous section, the numerical experiments for Minimal 1 are, in fact, done in this manner.

VI. CONCLUSIONS AND FUTURE 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 results show that the proposed approach achieves superior performance over the minimal approaches. More importantly, the proposed approach generalizes easily to a variety of 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 to work with cyclic graphs or to allow for semi-decentralized operation with only a limited amount of feedback. Also, more recent GA techniques, e.g., linkage learning GA which offers improved scalability by exploiting the correlations between variables that are to be learned as the algorithm progresses, are worth investigating 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] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *J. ACM*, vol. 27, no. 4, pp. 701–717, 1980.

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

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

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

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

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

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

[22] 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.

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

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

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