This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

An Evolutionary Approach To Inter-Session Network Coding

Minkyu Kim, Muriel Médard, Una-May O'Reilly, Danail Traskov

***Abstract*—Whereas the theory and application of optimal network coding are well studied for the single-session multicast scenario, there is no known optimal network coding strategy for a more general connection problem where there are more than one session and receivers may demand different sets of information. Though there have been a number of recent studies that demonstrate various utilities of network coding in the multi-session scenario, they rely on very restricted classes of codes in terms of the coding operations allowed and/or the location of decoding. In this paper, we propose a novel inter-session network coding strategy for a general connection problem. Our coding strategy allows fairly general random linear coding over a large finite field, in which decoding is done at receivers and the mixture of information at interior nodes is controlled by evolutionary mechanisms. We demonstrate how our coding strategy may surpass existing end-to-end pairwise XOR coding schemes in terms of effectiveness and practicality.**

I. Introduction

In the multicast scenario with a single session, the theory of network coding is well founded on the famous theorem by Ahlswede et al. [1] that characterizes the capacity region by the max-flow (min-cut) bounds. Subsequently, it is shown that the optimal capacity can be achieved using only scalar linear network codes [2]. However, when we generalize the problem such that there are more than one session and receivers may demand different sets of information, finding the optimal network coding strategy is still an open question.

First of all, in such a generalized problem, characterizing the capacity region becomes prohibitively difficult and even its inner/outer bounds cannot be computed in practice [3]–[6]. Moreover, linear coding is shown to be insufficient for optimal coding in the multi-session case [7]. A graph theoretic approach is proposed as a systematic method for deciding solvability of a given network with either linear or nonlinear codes [8], whose scalability issue, however, is unresolved. Even within linear codes, the solvability decision problem is shown to involve Gröbner basis computation [9], whose complexity may prohibit practical implementations for large problems. Recently, [10] suggests that solving a general network coding problem is equivalent in terms of complexity to solving a set of polynomial equations.

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).

U.-M. O'Reilly is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA (unamay@csail.mit.edu).

D. Traskov is with the Institute for Communications Engineering, Technical University Munich, Munich, Germany (danail.traskov@tum.de).

In short, it is very unlikely that a network coding strategy that is theoretically optimal will soon emerge. Nevertheless, there have been a number of studies that demonstrate various utilities of network coding in the multi-session scenario, based on some restricted classes of codes in terms of the coding operations allowed and/or the location of decoding. Katti et al.'s opportunistic coding [11] leads to a restricted, yet very practical coding scheme, taking advantage of the broadcast nature of wireless medium. In their coding scheme, multiple, possibly more than two, packets can be combined using binary XOR, but decoding needs to be done immediately at the neighbors of the coding node.

If we wish to perform decoding at receivers, we may need to put an even stronger restriction: XOR operations are allowed only between two flows, thus called *pairwise* XOR coding. The simplicity of coding and decoding operations of pairwise XOR codes allows the code construction problem to be described as flow formulations, which thus can be solved jointly with various other network flow problems. Traskov et al. [12] present linear and integer optimization formulations for the class of pairwise XOR coding for multiple unicast sessions. Ho et al. [13] develop back pressure algorithms for finding approximately throughput-optimal network codes within the class of pairwise XOR coding. Eryilmaz et al. [14] propose a dynamic scheduling strategy that supports the achievable rates for multiple unicast sessions given in [12]. Note, however, that the benefit of pairwise XOR coding within that framework presented in [12], in terms of savings in link cost compared with traditional routing, is found to be only modest depending on network topologies. It is not clear whether such a modest gain is because the simulations performed in [12] were restrictive, or it actually indicates the limitations of the pairwise XOR coding.

Even if we may wish to lift the restriction on the number of sessions that can be coded together or the type of coding operation, it is hard to do so because of the lack of appropriate tools for that. The flow formulations in [12] or [13] may be generalized to represent a more general XOR coding scheme that allows coding among more than two sessions. However, keeping track of all possible combinations of coded streams among up to k sessions would require at least $O(m^k|E||V|)$ variables, where m is the total number of sessions, which leads to a prohibitively large number of constraints, hindering practical implementations. Beyond the binary field, a path-based characterization is presented for the feasibility of the connection problem with two unicast sessions [15], based on which a distributed rate control algorithm is proposed

978-1-4244-3513-5/09/$25.00 ©2009 IEEE

[16]. However, a generalized characterization in the case of coding among more than two sessions still seems difficult. For more general coding schemes than the pairwise XOR that allow decoding at receivers, a linear optimization problem is proposed by Lun et al. [17], whose minimum cost is shown to be no greater than the minimum cost of any routing solution. However, its formulation is based on the given set of the source processes that can be mixed on each link, which still remains difficult to decide optimally.

In this paper, given that there exists a wide gap between the presumably difficult quest for optimal inter-session network coding and many existing approaches that are constrained within very restrictive classes of coding strategies, we investigate how the evolutionary approaches based on a Genetic Algorithm (GA), which in our previous works [18]–[22] have been used for the network coding resource optimization problem, can be utilized in an effort to fill the gap. In particular, we present a novel randomized linear coding scheme, in which decoding happens at receivers while interior nodes perform random linear coding with selective mixture of information, controlled by evolutionary mechanisms. We then demonstrate, through simulations, how our coding strategy may surpass existing end-to-end XOR coding schemes in terms of effectiveness and practicality.

The rest of the paper is organized as follows. Section II describes the problem setup with a related background. Section III presents our coding strategy with a brief introduction to GA. Section IV describes how the computational components of our evolutionary approach need to be designed, and then Section V shows how our approach can be implemented in a distributed fashion over the network. Section VI exhibits a number of simulations for the evaluation of the performance of our coding strategy. Section VII concludes the paper with some directions for future research.

II. Problem Setup and Background

A. Problem Setup

We assume that the network is given by an acyclic directed multigraph $G = (V, E)$ where each link has a unit capacity and links with larger capacities are represented by multiple links. Only integer flows are allowed, hence there is either no flow or a unit rate of flow on each link.

Let us assume that there are R independent random processes of unit entropy rate, denoted by $\mathcal{X}$=$\{X_1, X_2, ..., X_R\}$, originating at $s(\geq 1)$ source nodes. There are d receiver nodes, $v_1, v_2, ..., v_d$. For each receiver v_i $(i = 1, ..., d)$, we denote the set of requested source processes by $\mathcal{X}_i \subset \mathcal{X}$. We assume that for each source process in $\mathcal{X}_i$ $(i = 1, ..., d)$, there exists a path from its originating node to receiver v_i; otherwise, it is easy to check and declare that the problem is not solvable. Other than this connectivity condition, we do not put any restriction on the source processes each receiver node may request, i.e., $\mathcal{X}_i$ can be any nonempty subset of $\mathcal{X}$ provided that receiver v_i is reachable from the originating nodes of the source processes in it.

For this generalized scenario, there is no simple characterization known for the feasibility of the connection problem with network coding, even for the simpler case of unicast connections, i.e., when all $\mathcal{X}_i$'s are disjoint [9], [23]. Though linear coding has been shown to be suboptimal in general [7], here we focus on the scalar linear coding in a general form, i.e., we do not restrict ourselves within either binary XOR operations or pairwise mixing.

Because the feasibility characterization still remains hard even within linear coding [9], we assume that the capacity constraints can be relaxed for some links, i.e., links can be scaled up, if necessary, to allow multiple transmissions. Note that this may be the case in many practical networking scenarios; e.g., in wireless networks, nodes can exploit more capacities at the expense of more energy, and in optical networks, capacity itself is rarely a limited resource, though using more resources incurs an additional cost. With this assumption, the feasibility problem can be resolved by scaling links appropriately.

Then, we mainly focus on finding a cost-efficient transmission scheme using network coding. Hence, our primary objective is to minimize the link cost required to satisfy the given communication demands. Later, we consider another objective of minimizing the coding resources, i.e., number of coding nodes/links.

B. Background

For the multicast case, the solvability of a connection problem boils down to whether the max-flows between the source node and each of the receiver nodes all exceed the desired multicast rate [1]. This can be translated into the algebraic framework [9] such that, if we let $\underline{\xi}$ denote the vector consisting of all the link coefficients as defined in [1], the problem of finding a feasible network code becomes finding an assignment of numbers to variables $\underline{\xi}$ such that the product of the determinant polynomials does not evaluate to zero, which can be done relatively easily with many efficient randomized algorithms, e.g., as in [24].

As we go beyond the multicast case, we now have another condition for solvability: within the transfer matrix, the submatrices relating the input processes that are not requested to the corresponding output processes at the receiver nodes should evaluate to zero (the first condition in Theorem 6 in [9]). Let us denote the *entries* of the submatrices that have to evaluate to zero by $f_1(\underline{\xi}), f_2(\underline{\xi}), ..., f_K(\underline{\xi})$, which we refer to as *interference polynomials*. As in the multicast case, we still have the condition that the submatrices associated with the input and output processes specified in the connection requests should be nonsingular (the second condition in Theorem 6 in [9]). Let $g_1(\underline{\xi}), g_2(\underline{\xi}), ..., g_L(\underline{\xi})$, referred to as *determinant polynomials*, be the determinants of the submatrices that should be nonsingular.

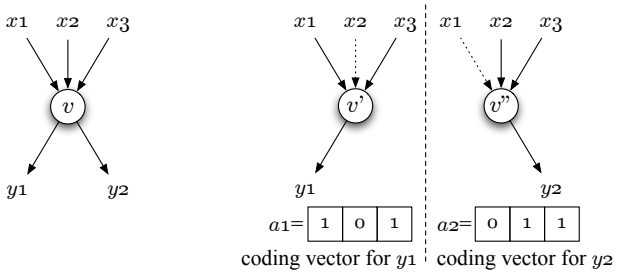
While finding an assignment that makes determinant polynomials nonzero can be done easily, the difficult part is to solve the set of determinant polynomials to find an assignment of numbers to $\underline{\xi}$ that makes them all zero. To determine whether

such an assignment is possible, [9] suggests a Gröbner basis method for determining the emptiness of variety of the ideal generated by the interference polynomials and a function of the determinant polynomials. However, the worst case complexity of the Buchberger algorithm, which is used predominantly to compute a Gröbner basis, is doubly exponential [25] and moreover, in our case the input to the Buchberger algorithm, i.e., the interference and determinant polynomials, may already have an exponential number of terms, and hence calculating a Gröbner basis may not be a practical solution anyway. On the other hand, as solving a general linear network coding problem is no easier than solving a set of polynomial equations [10], it is unlikely that one can develop a specialized method that possibly provides more efficiency, utilizing some structural properties of the interference and determinant polynomials.

III. Our Coding Strategy: Selective Random Linear Coding

In our coding strategy, we control the mixture of information at each node by deciding whether to include the input from a particular incoming link when calculating the output on each outgoing link. However, once the decisions are made, we calculate the output by forming a random linear combination of the inputs allowed. Hence, the name *selective random linear coding*. Intuitively, the strategy we employ to deal with the interference polynomials is 1) to make some interference polynomials identically zero by zeroing out an enough number of associated components of $\underline{\xi}$ and 2) for the interference polynomials that remain nonzero, to allow enough degrees of freedom at the receivers so that the unwanted information can be successfully canceled out.

For a node v with d_{in} incoming links and d_{out} outgoing links, we assign a binary variable a_{ij} to each pair of the $i \in \{1, ..., d_{in}\}$-th incoming link and the $j \in \{1, ..., d_{out}\}$-th outgoing link. For the j-th $(j = 1, ..., d_{out})$ outgoing link, we refer to the associated binary variables $a_j = (a_{ij})_{i \in \{1,...,d_{in}\}}$ as a *coding vector* (see Fig. 1 for an example). The coding vectors are the variables that we need to decide in our coding strategy.



(a) Node v (b) Coding vectors for outgoing links

Fig. 1. Node v with 3 incoming and 2 outgoing links is associated with two coding vectors $a_1 = (a_{11}, a_{21}, a_{31})$ and $a_2 = (a_{12}, a_{22}, a_{32})$.

Once a set of the coding vectors is given, we employ random linear coding at interior nodes, selectively using only those inputs associated with 1's in the coding vectors. More specifically, node v calculates the output y_j $(j = 1, ..., d_{out})$ on the j-th outgoing link as follows. Define the set I_j of indices as

$$I_j = \{1 \le i \le d_{in} | \text{ the } i\text{-th component of } a_j \text{ is } 1\}.$$

If we denote the input from the i-th $(i = 1, ..., d_{in})$ incoming link by x_i,

$$y_j = \sum_{i \in I_j} \text{rand}(\mathbb{F}_q) \cdot x_i$$

where $\text{rand}(\mathbb{F}_q)$ denotes a nonzero random element from $\mathbb{F}_q$. If the set I_j is empty, y_j is assumed to be zero.

Given a set of coding vectors, the feasibility verification can be done by first performing selective random linear coding at interior nodes as described above. Then, each receiver node v_i $(i = 1, ..., d)$ performs Gaussian elimination to determine whether all the desired input processes can be recovered, with the interference part canceled out. Note that this whole process can be done in a distributed manner, which will be utilized later.

This randomized decision rule incurs an error when nonzero polynomials evaluate to zero after randomly assigning values to variables; for zero polynomials, random assignments in the evaluation process do not affect the final result. Hence, the error probability is bounded by $1 - (1 - d/q)^{\nu}$ where q is the size of the finite field used for coding and ν is the maximum number of links in any set of links constituting a flow solution from the source to any receiver [24]. Note that this bound remains the same even if we scale up some of the links as will be discussed later.

It remains to find an optimal assignment of the coding vectors out of a exponentially scaling number of possible choices, which we address using a GA based search method. Before proceeding, let us provide a brief introduction to GA.

A. A Brief Introduction to GA

GAs are stochastic search methods that mimic genetic phenomena such as gene recombination, mutation and survival of the fittest. Having been applied to a large number of scientific and engineering problems, GAs are especially shown to be effective for the problem of network coding resource optimization [18]–[22]. The main control flow of the standard form of GA, called *simple GA*, is shown in Fig. 2 [26].

Simple GA [26] operates 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 chromosomes that are more fit are selected with higher probability. For crossover, chromosomes are randomly paired, and then two chromosomes in each pair exchange a subset

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

Fig. 2. Main control flow of simple GA.

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 simple GA. The above process is iterated with the newly generated population successively replacing the current one. Simple GA terminates when a certain stopping criterion is reached, e.g., after a predefined number of generations. For further details of a standard simple GA, the reader is referred to [18], [26].

IV. Computational Aspects

We first describe how the computational components of our evolutionary approach need to be designed, and then in the next section, we present how our approach can be implemented in a distributed manner over the network.

A. Chromosome and Fitness Function

The decision variables in our coding scheme are a set of coding vectors as defined above in Section III. Hence, each chromosome (i.e., a candidate solution) consists of the collection of all coding vectors. For a given chromosome $\underline{y}$, we must verify its feasibility first by performing selective random linear coding as described in Section III. Once the feasibility test is done, given that our primary objective is to obtain the cost-efficient transmission scheme via network coding, the fitness function F is defined as

$$F(\underline{y}) = \begin{cases} \text{total cost of link usage}, & \text{if } \underline{y} \text{ is feasible}, \\ \infty, & \text{if } \underline{y} \text{ is infeasible}. \end{cases} \quad (1)$$

Note that if we wish to minimize the resources engaged in network coding, we may replace the link cost by the number of coding nodes/links for feasible chromosomes. Moreover, the two objective values, i.e., the link and coding costs, can be jointly considered to investigate a possible tradeoff between those two objectives [22], which leads to more informed decisions on whether or where to employ network coding.

B. Initial Population Construction

Typically the initial population of a GA is composed of random chromosomes. However, as pointed out in [18], inserting some non-random chromosomes can greatly improve the performance of the algorithm. When we add a number of non-random solutions to the initial population, care must be taken to insert only *neutral* solutions in the sense that they are not particularly close to some local optimum. Otherwise, those inserted starting points tend to take over the whole population in the early stage of the evolution process so that the algorithm may end up converging just to a neighborhood of the starting points.

To create a neutral starting point, we first scale up the links in G to make the problem multicast-like in the sense that each receiver node now receives all source processes that have a directed path to it. Then, we employ as a neutral starting point the all-one chromosome that indicates mixing everything at all interior nodes.

Let us now show how we can find an appropriate scaling factor. Before proceeding, for each receiver node v_i $(i = 1, ..., d)$ in G, we let $\mathcal{Y}_i \subset \mathcal{X}$ be the set of the source processes from whose originating node, receiver v_i is reachable (i.e., there exists a directed path from the source process's originating node to node v_i). Also, we let r_i $(i = 1, ..., d)$ denote the size of set $\mathcal{Y}_i$. Now let us create an auxiliary network H from G by introducing a virtual source node S and adding a unit-capacity link from node S to the source node at which each source process $X_j \in \mathcal{X} (j = 1, ..., R)$ originates. Then, we let f_i $(i = 1, ..., d)$ be the value of the maximum flow from the virtual source node S to each receiver v_i $(i = 1, ..., d)$. We define k as

$$k = \left\lceil \max_{i \in \{1,...,d\}} \left(\frac{r_i}{f_i} \right) \right\rceil. \quad (2)$$

Theorem 1: Let G' be a scaled version of G with each link replaced by k multiple links, where k is defined as in (2). In network G', a network code that performs random linear coding using all available inputs at every interior node in a sufficiently large finite field is feasible.

Proof Outline: Let us create another auxiliary network H' by scaling up the links in H by a factor of k and adding unit-capacity links between each source process in $\mathcal{X} \setminus \mathcal{Y}_i$ and each receiver node v_i $(i = 1, ..., d)$. Then, we have a multicast problem for which a feasible network code can be obtained by employing random linear coding in a sufficiently large finite field. Given such a feasible network code $\mathcal{C}$ on network H', it can be shown that $\mathcal{C}$ remains feasible even if we remove the added links to obtain network G'. (A full proof can be found in [27].) □

Note that the all-one chromosome shown above is feasible but has the worst cost in terms of either the link cost or the coding cost and thus it may not bias the initial population toward any particular suboptimal solution.

C. Genetic Operators

In conventional crossover and mutation operators, which we refer to as *bit-wise* operators, the unit of interchanged or perturbed subcomponents is each *bit* of the chromosomes. However, it is pointed out in [19], [20] that, for the problem of the coding resource optimization for multicast, a significant

performance gain can be attained by applying the crossover and mutation operations on each *full coding vector*. That is, for vector-wise crossover, we let two chromosomes subject to crossover exchange each full coding vector (rather than each bit) independently with the given crossover probability. For vector-wise mutation, we randomly regenerate each coding vector (again, rather than each bit) independently with the given mutation probability. We refer to these two operators as *vector-wise* operators.

Given that enforcing coding vector-level modularity in genetic operations can lead to a significant performance gain, we may develop another set of genetic operators that further exploit, this time, the node-level modularity. In *node-wise* operators, we now interchange or perturb all coding vectors associated with the outgoing links of *each node* with the crossover/mutation probability. The intuition behind the node-wise genetic operators is that the coefficients of the links directly connected with each other or within a few hops away are more likely to have strong dependencies than those associated with the links far away from each other. Note, however, that the node-wise genetic operators enforce a broader level of modularity than the vector-wise operators, exchanging or perturbing a larger number of coding vectors at once. If this is the right level of modularity, it would translate into faster convergence of the algorithm to the same or a better solution; otherwise, the algorithm may tend to converge prematurely to a lower quality solution.

We will later verify through simulations the effect of these different genetic operators on the performance of the algorithm.

V. Distributed Implementation

Recall that in Section III we discussed how the feasibility test of a single chromosome can be done by employing random linear coding at interior nodes. Note that in doing so, each interior node only refers the relevant portion of the chromosome, i.e., the coding vectors that indicate the operations at that node. Hence, we can divide up the population by letting each node handle only the coding vectors it needs from every chromosome in the population.

If the coding vectors are stored at local nodes, we only need to transmit the set of the coefficients that indicate the overall effect of network coding relative to the source data, which is commonly referred to as a *global encoding vector* in the network coding literature (see e.g., [28]), for feasibility test. Hence, feasibility test can be done in a distributed fashion by transmitting packets containing such coefficients.

Moreover, it can be shown that the whole fitness calculation and all genetic operations discussed in Section IV can be done independently at local interior nodes with some coordination information embedded in data packets, assuming that the source nodes can communicate with one another [27]. Note that when we consider network coding among multiple sessions that may originate from multiple source nodes, it must be assumed in the first place anyway that the source nodes can communicate with one another to find out the total number of the source processes being considered together for possible coding so that the coefficients for linear coding can be aligned consistently across all the source nodes involved. Hence, we assume that among the participating source nodes, we can designate one as the *master node* which serves as the main controller of the algorithm by gathering information from other source nodes and sending the calculated coordination information to other source nodes.

With the above assumptions, the whole evolutionary algorithm to search for an optimal set of coding vectors can operate in a distributed fashion [27], whose overall flow is shown in Fig. 3 with the location of each procedure specified. More detailed descriptions of the distributed algorithm are omitted for space consideration (the reader is referred to [27, Chapter 7] for details). Note that the calculation of the scaling factor k defined in (2) can also be done in a distributed manner [27]. Computationally, the distributed algorithm performs the same task as simple GA (Fig. 2) with the computational components described in Section IV.

```
[S1]  initialize; (all nodes)
[S2]  run forward evaluation phase; (all nodes)
[S3]  run backward evaluation phase; (all nodes)
[S4]  send partial fitness to master node; (source nodes)
[S5]  calculate fitness; (master node)
[S6]  while termination criterion not reached (master node)
      {
[S7]    calculate coordination vector; (master node)
[S8]    fetch coordination vector from master node; (source nodes)
[S9]    run forward evaluation phase; (all nodes)
[S10]   perform selection, crossover, mutation; (interior nodes)
[S11]   run backward evaluation phase; (all nodes)
[S12]   send partial fitness to master node; (source nodes)
[S13]   calculate fitness; (master node)
      }
```

Fig. 3. Flow of distributed algorithm for selective random linear coding

Within this distributed setup, the fitness evaluation of each chromosome requires the computational complexity of $O(d_{in} \cdot d_{out} \cdot R)$ at each interior node and $O({d_{in}}^2 R)$ at each receiver node [27].

The most important benefit of this distributed structure is that a network code can be constructed on the fly while the network is operational, allowing for the following network coding protocol: As the master node sends a packet that signifies the start of the code construction, all participating nodes go into the code construction mode, running the algorithm described above. As the distributed evolutionary algorithm proceeds, each interior node stores and improves its relevant network codes. At the end of the algorithm, the master node only needs to send the index of the best chromosome of the last population and all participating nodes now start to transmit data based on the locally stored network code that corresponds to the received best index.

VI. Performance Evaluation

In order to evaluate the effect of different genetic operators and also compare the performance of our coding scheme with others, we first perform simulations for the multiple-unicast scenario, for which other network coding schemes are available. Then, to demonstrate our algorithm's ability to handle more general problems, we further consider the case of wireless networks with no restriction on the type of connection requests.

A. Multiple Unicast Connections

We performed a number of simulations for the multiple-unicast case, i.e., all connection requests $\mathcal{X}_i$'s $(i = 1, ..., d)$ are disjoint. Our simulations are based on the grid network introduced in [12] (network D in Fig. 4). In [12], the cost of each link is assigned randomly, which allows only slim chances that the network coding advantage exists. Note, however, that network coding gain takes effect only if there exists an expensive bottleneck link that has to be used by a number of flows and also lower cost detours around it, which happens rarely when the connection requests and link costs are randomly chosen. Hence, in our experiments, we pick a cost assignment as depicted in Figure 4, where the links with a cost higher than 1 are highlighted by thicker arrows with the actual cost shown by their side, to make the network coding advantage clearly exist at least for some connection requests.

For comparison, we take Traskov et al.'s pairwise XOR coding scheme [12]. We do not include the approach by Wang et al. [15] here because it can be shown that the grail structure considered therein does not lead to link cost savings in our problem setup (i.e., our primary objective to minimize the link cost whereas the capacity constraint may be relaxed if needed); nevertheless, it does increase the capacity region when the capacity constraint is strictly enforced.

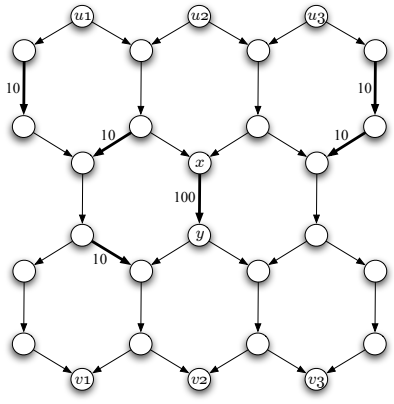


Fig. 4. Grid network D for multiple-unicast simulations

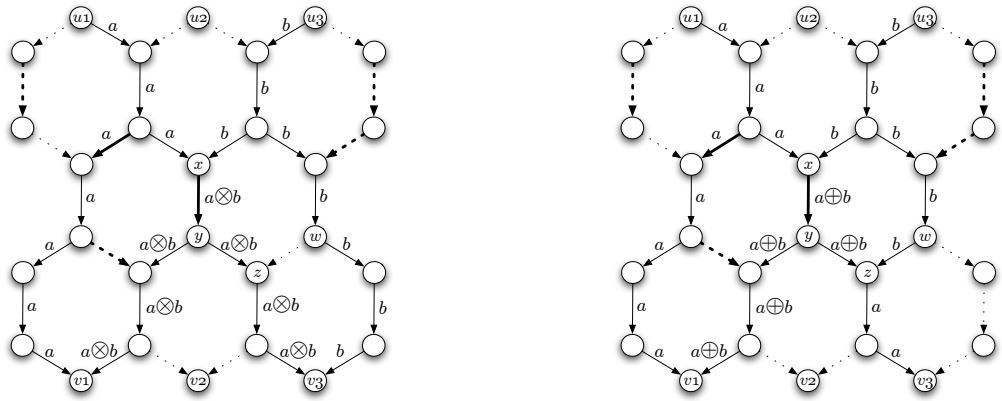
1) Two-Connection Case: First, we consider the case of two connections, i.e., $R = 2$. We repeatedly ran our algorithm, varying the location of the source processes and connection requests such that two source processes X_1 and X_2 may originate at any of nodes $\{u_1, u_2, u_3\}$ and each source process may be requested at any of receiver nodes $\{v_1, v_2, v_3\}$. For comparison with the routing case and the pairwise XOR coding scheme, we used the multi-commodity formulation with integer constraints and Traskov et al.'s formulation [12], respectively.

Among all possible arrangements of the source processes and connection requests, there is only one case in which network coding saves the link cost: X_1 and X_2 originate at nodes u_1 and u_3, respectively, while X_1 and X_2 are requested at nodes v_3 and v_1, respectively. Fig. 5(a) shows the best solution obtained by our selective random linear coding which requires the link cost of 131. Note that, for this simple problem, our evolutionary algorithm, despite its stochastic nature, always yields the same solution regardless of the genetic operators used. In comparison, the optimal cost achieved by routing is 212. In Fig. 5(a), we use the symbol $\otimes$ to represent linear combination with random coefficients from the designated finite field; i.e., $x_1 \otimes x_2 \otimes ... \otimes x_n = \sum_{i=1}^{n} \text{rand}(\mathbb{F}_q) \cdot x_i$, where again $\text{rand}(\mathbb{F}_q)$ represents a nonzero random element from the designated finite field $\mathbb{F}_q$. Interestingly, this example highlights the difference between our coding scheme and the pairwise XOR coding scheme, whose best solution, as depicted in Fig. 5(b), offers an even lower cost of 129. In our selective random linear coding, coding operation is performed only once at node x and decoding is done at receiver nodes. On the other hand, in the best pairwise XOR code, another XOR operation is done at node z, which, in fact, serves as decoding at an interior node and consequently saves the link cost of 2 (by not sending β at node w along a longer path down to node u). Note that the same transmission strategy would not work for our selective random linear coding because another coding operation at node z would yield another random linear combination $(a \otimes b) \otimes b$ rather than a.

From this example, we observe that, while both coding schemes provide advantages over traditional routing, our selective random linear coding scheme may require some additional link cost compared with the pairwise XOR coding, which can be considered the price of employing randomized coding in a finite field larger than the binary field. Note, however, that as we increase the number of connections, our coding scheme leads to a much more practical solution despite the possible expense of such additional link cost, as will be discussed next.

2) Five-Connection Case: Let us now increase the number of source processes to 5 and pick up an arrangement of the source processes and connection requests that allows the network coding advantage: {(source process, source node, receiver node)} = $\{(X_1, u_1, v_3), (X_2, u_1, v_1), (X_3, u_3, v_1), (X_4, u_3, v_2), (X_5, u_3, v_2)\}$.

As opposed to the simpler problem in the previous subsection, the performance of our algorithm varies depending on the type of genetic operators used. Table I summarizes the performance of our algorithm with different genetic operators. For each type of genetic operators, we performed 30 simulation runs and at the end of each run we pick the best coding solution out of the last population. The first column shows the lowest cost of those 30 best coding solutions and the next



(a) Selective random linear coding

(b) Pairwise XOR coding

Fig. 5. Comparison of best coding solutions obtained.

two columns show the average and standard deviation of the cost of those 30 solutions. For comparison, the best routing solution yields the optimal cost of 242. The fourth column of the table displays the ratio of the algorithm runs, out of the total of 30, in which the cost of the best coding solution found is actually lower than the best routing solution. The fifth column calculates the average number of generations required for a coding solution that outperforms the routing solution, if found, to appear in the population. The last column shows the ratio of the algorithm runs in which a coding solution outperforming the routing solution is found before the 100-th generation, relative to the total number of algorithm runs where such a coding solution is ever found.

TABLE I
SUMMARY OF THE LINK COSTS OF THE CODING SOLUTIONS FOUND BY DIFFERENT GENETIC OPERATORS AND RELATED STATISTICS. THE BEST ROUTING SOLUTION REQUIRES THE LINK COST OF 242.

	Link Cost			Outperform Routing		
	Best	Avg	Std	Ratio	At	<100
Bit-wise	181	204.2	10.52	1.00	296.7	0.07
Vector-wise	156	164.8	5.87	1.00	177.3	0.73
Node-wise	158	236.2	41.96	0.43	124.8	0.77

From Table I, we first notice that our algorithm with the bit-wise or vector-wise operators reliably yields a network coding solution that outperforms the best routing solution. Between the two kinds of operators, the vector-wise operators lead to much better solutions, both in terms of the mean and standard deviation. Also, the number of generations required to find a network coding better than the best routing solution is much smaller on average for the vector-wise operators; most of the time it is found before the 100-th generation. Hence, we may conclude that the vector-wise operators allow the algorithm to find much better solutions much faster than the bit-wise operators.

For the case of the node-wise operators, our algorithm finds a network coding solution that exceeds the routing solution only in 43% of the simulations. However, in those successful simulations, such network coding solution is found much faster than the case of the vector-wise operators. Hence, we may conclude that with the node-wise operators the algorithm tends to converge faster, but prematurely to a lower quality solution. This may be due to that the node-level modularity enforced by the node-wise operators is too strong in the sense that it changes too many coding vectors at once, which conceptually may correspond to setting the step too large in an iterative optimization scenario.

On the other hand, the optimization formulations for the pairwise XOR coding [12] failed to converge within a reasonable amount of time (during a full week of simulations) based on the simulation environment used in [12]. Note that the linear and integer programs in [12], even for this five-connection problem, contain around 68700 and 1400 variables (including the slack variables to handle the $\max$ operator in the constraints) and 67500 and 1700 constraints, respectively. Though we may have been able to obtain converged results if we had experimented it with a much faster machine, the point we would like to make here is that the optimization formulations considered in [12] may not provide a practical scalability as the number of connections increases. For comparison, our algorithm takes about 1.5 seconds for each generation in the same simulation environment, where we simulated each node's operation sequentially (one at each time from upstream to downstream nodes) using MATLAB on a single-processor machine. In a real network, where each node uses its own computational resources, we may expect much faster execution of our algorithm.

B. General Connections

We performed another set of experiments in fully generalized networking scenarios where the receiver nodes may request any combination of the available source processes. For this, we generated 100 random wireless networks where 40 nodes are placed randomly within a 10×10 square with radius of connectivity 3. A unit-rate hyperlink is originated from each node toward the set of nodes that are within the

connectivity range and have a higher horizontal coordinate. In each random topology, 5 source processes were randomly placed at 5 nodes chosen in the left half and each of 5 receiver nodes randomly chosen in the right half demands a randomly chosen nonempty subset of $\mathcal{X} = \{\mathcal{X}_1, ..., \mathcal{X}_5\}$.

For such generated random connection problems, we apply a two-stage method in which we first run our algorithm with the fitness function defined as (1) to minimize the link cost without restricting network coding. Then, from the best coding solution found, we take only the links that are used for transmission to form the subgraph to be used in the second stage. In the second stage, we use our algorithm to minimize the number of coding nodes by changing the fitness function to reflect the number of nodes where network coding is performed.

Table II shows the distribution of the found minimum number of coding nodes through the two-stage method. Note that in most cases (about 80% of the random topologies tested), the calculated minimum cost can be achieved without network coding at all. Also, for the topologies where network coding is required, it is required only at a small subset of nodes, i.e., not all nodes need to perform network coding, which was also the case in many multicast scenarios [27].

TABLE II
DISTRIBUTION OF THE CALCULATED MINIMUM NUMBER OF CODING NODES IN 100 RANDOM TOPOLOGIES.

# Coding Nodes	0	1	3	4	5	6	7	9	$\geq$10
# Topologies	79	6	1	1	1	3	2	2	5

C. Discussion

Above simulations exhibit that our evolutionary approach offers a coding strategy in the general connection problem which is still somewhat restricted but with enhanced features in many aspects, compared with existing pairwise XOR coding. For the problem of multiple unicast connections, we showed that our evolutionary approach approach yields a network coding solution that offers an advantage over traditional routing in terms of link cost, whereas an existing approach for pairwise XOR coding may fail to provide a scalability within practical ranges as the number of connections increases. Also, we demonstrated that our evolutionary approach can tackle more general problems in which there is no restriction on the type of connection requests, taking the coding cost into account as well. Note that, though only the two-stage method was experimented for an illustrative purpose in the second set of simulations, a multi-objective evolutionary approach can also be utilized, similarly as in [22], to investigate the tradeoff between the two objectives. In addition to the distributed structure presented in Section V, another distributed structure, i.e., temporally distributed structure, can also be adopted for more efficient utilization of computational resources as well as more robust operation against packet losses [21].

A possible drawback of our evolutionary approach is that, as opposed to the case to the coding resource optimization for multicast [19], it lacks a performance bound. However, given that there are no practical alternatives that take into account coding among more than two flows, our approach may serve as a unique means for exploring network coding advantages in a much more generalized setup than the pairwise coding. Another possible limitation is regarding the scalability; i.e., as is typical for a GA, the population size may need to be increased significantly for large problems. However, with the distributed structure described in Section V, the population size can be increased rather flexibly by increasing the size of the packet used for fitness evaluation, given that the computational complexity at each node scale linearly with the population size. Moreover, with the temporally distributed structure [21], we may further increase the effective population size by increasing the number of (sub)populations.

VII. CONCLUSION AND FUTURE WORK

We have proposed a novel inter-session network coding strategy for a general connection problem beyond multicast, for which no optimal network coding scheme is known. Our coding strategy allows random linear coding over a large finite field, in which decoding is done at receivers and the mixture of information at interior nodes is controlled by evolutionary mechanisms. We have demonstrated how our coding strategy may surpass existing end-to-end pairwise XOR coding schemes in terms of effectiveness and practicality.

There have been many recent developments in the field of evolutionary computation that significantly improve the scalability of the traditional simple GA, on which our current approach is based. Such advanced GA frameworks can be readily employed, without changing the overall framework presented here, to find and exploit further linkage information, i.e., dependencies among variables, for an improved performance. Also, one may develop a method to construct solutions that provide some useful bounds on the link or coding cost, which in turn can be combined with our evolutionary approach to generate a better initial population or to refine the final solution (as in the coding resource optimization problem for multicast [19]). In addition, one may utilize the core structure of our proposed evolutionary approach within other network problems that involve combinatorial optimizations with an unresolved scaling issue. For instance, for a problem in which the number of variables/constraints scales exponentially with the size of the network, an evolutionary algorithm may effectively be utilized to yield a more compact set of variables/constraints that needs to be considered at a time.

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, 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, 2003.

[3] R. W. Yeung, S.-Y. R. Li, N. Cai, and Z. Zhang, "Network coding theory part II: Multiple source," *Foundation and Trends in Communications and Information Theory*, vol. 2, no. 5, pp. 330–381, 2005.

[4] R. W. Yeung, *A First Course in Information Theory,*. Kluwer Academic/Plenum Publishers, 2002.

[5] T. Chan and A. Grant, "Dualities between entropy functions and network codes," *submitted to IEEE Trans. Inform. Theory (arXiv:0708.4328v1)*, 2007.
[6] X. Yan, R. Yeung, and Z. Zhang, "The capacity region for multi-source multi-sink network coding," in *Proc. IEEE ISIT*, 2007.
[7] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow," *IEEE Trans. Inform. Theory*, vol. 51, no. 8, pp. 2745–2759, 2005.
[8] J. K. Sundararajan, M. Médard, R. Koetter, and E. Erez, "A systematic approach to network coding problems using conflict graphs," in *Proc. Information Theory and its Applications*, 2006.
[9] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 782–795, 2003.
[10] R. Dougherty, C. Freiling, and K. Zeger, "Linear network codes and systems of polynomial equations," *IEEE Trans. Inform. Theory*, vol. 54, no. 5, pp. 2303–2316, 2008.
[11] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Médard, "The importance of being opportunistic: Practical network coding for wireless environments," in *Proc. Allerton Conference*, 2005.
[12] D. Traskov, N. Ratnakar, D. S. Lun, R. Koetter, and M. Médard, "Network coding for multiple unicasts: An approach based on linear optimization," in *Proc. ISIT*, 2006, pp. 1758–1762.
[13] T. Ho, Y.-H. Chang, and K. J. Han, "On constructive network coding for multiple unicasts," in *Proc. Allerton Conference on Communication, Control and Computing*, 2006.
[14] A. Eryilmaz and D. S. Lun, "Control for inter-session network coding," MIT-LIDS, Tech. Rep. P-2722, 2006.
[15] C.-C. Wang and N. B. Shroff, "Beyond the butterfly - graph-theoretic characterization of the feasibility of network coding with two simple unicast sessions," in *Proc. IEEE ISIT*, 2007.
[16] A. Khreishah, C.-C. Wang, and N. B. Shroff, "An optimization based rate control for communication networks with inter-session network coding," in *Proc. IEEE Infocom*, 2008.
[17] D. S. Lun, M. Médard, T. Ho, and R. Koetter, "Network coding with a cost criterion," MIT-LIDS, Tech. Rep. P-2584, 2004.
[18] M. Kim, C. W. Ahn, M. Médard, and M. Effros, "On minimizing network coding resources: An evolutionary approach," in *Proc. NetCod*, 2006.
[19] M. Kim, M. Médard, V. Aggarwal, U.-M. O'Reilly, W. Kim, C. W. Ahn, and M. Effros, "Evolutionary approaches to minimizing network coding resources," in *Proc. IEEE Infocom*, 2007.
[20] M. Kim, V. Aggarwal, U.-M. O'Reilly, and M. Médard, "Genetic representations for evolutionary minimization of network coding resources," in *Proc. EvoWorkshops*, 2007.
[21] ——, "A doubly distributed genetic algorithm for network coding," in *Proc. ACM Genetic and Evolutionary Computation Conference (GECCO)*, 2007.
[22] M. Kim, M. Médard, V. Aggarwal, and U.-M. O'Reilly, "On the coding-link cost tradeoff in multicast network coding," in *Proc. MILCOM*, 2007.
[23] Z. Li and B. Li, "Network coding: The case of multiple unicast sessions," in *Proc. Annual Allerton Conference on Communication, Control, and Computing*, 2004.
[24] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.
[25] M. Milanič, *Report on DIMACS working group on data de-identification, combinatorial optimization, graph theory, and the stat/OR interface*, 2005.
[26] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1996.
[27] M. Kim, "Evolutionary approaches toward practical network coding," Ph.D. dissertation, MIT, 2008, (available online at http://web.mit.edu/minkyu/www/doc/PhDthesis.pdf).
[28] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. Annual Allerton Conference on Communication, Control, and Computing*, 2003.