

# ON THE CODING-LINK COST TRADEOFF IN MULTICAST NETWORK CODING

Minkyu Kim\*, Muriel Médard\*, Varun Aggarwal†, and Una-May O’Reilly†

\*Laboratory for Information and Decision Systems

†Computer Science and Artificial Intelligence Laboratory

Massachusetts Institute of Technology, Cambridge, MA 02139

Email: {minkyu@, medard@, varun\_ag@, unamay@csail.}mit.edu

**Abstract**—We investigate the issue of the tradeoff between network coding and link usage in multicast network coding. Network coding makes minimum-cost multicast, an NP-complete problem with traditional routing alone, polynomially solvable, but if we consider the network coding capability as a resource, the link cost is actually minimized at the expense of the coding cost. We show that identifying such a tradeoff is NP-hard. For this problem, we propose an evolutionary approach that generalizes our previously proposed algorithms for coding resource optimization. Based on an existing multi-objective genetic algorithm, we develop a novel selection mechanism that utilizes some specific characteristics of the problem. We then show that the algorithm can be implemented in a distributed manner and evaluate the algorithm’s performance by performing experiments on several topologies.

## I. INTRODUCTION

Network coding has been shown to allow for minimum-cost multicast, whereas with traditional routing alone, the problem to achieve minimum-cost multicast is NP-complete and only suboptimal approximation methods are available [1]. Therefore, network coding transforms a presumably intractable task to a polynomially solvable problem.

Network coding, however, may incur some additional cost such as computational overhead or transmission delay, and thus the network coding capability can be considered a resource subject to optimization. In essence, what one may achieve with the method in [1] is to get the cost of link usage minimized at the expense of the cost of network coding. It is pointed out in [2] that network coding advantage can often be achieved with performing coding operations only at a subset of nodes. In particular, though it is necessary to allow network coding at all possible nodes initially to calculate a minimum-cost subgraph, there may be very few nodes in the resulting subgraph where network coding is actually required. Thus, as illustrated in [2], a two-stage method, where we first assume network coding everywhere and then try to minimize the number of coding nodes/links on the resulting minimum-cost subgraph, may be an effective tool to achieve minimum-cost multicast.

However, if network coding becomes necessary at some nodes after we constrain the information flow onto a selected subgraph, one may be interested in finding whether providing some extra capacities would eliminate the requirement of network coding, i.e., there may be a possible tradeoff opportunity

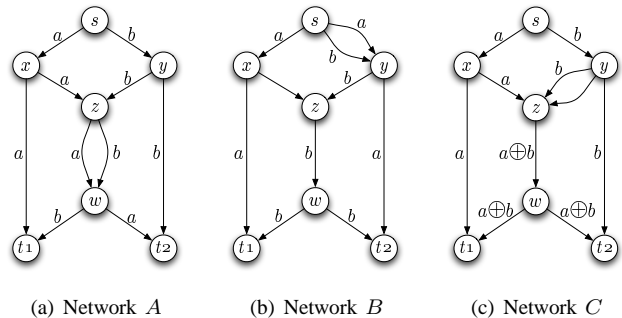


Fig. 1. Sample Networks for Example 1

between the coding and link costs. Consider the following example where each link has a unit capacity and a unit cost, and the desired multicast rate from  $s$  to  $t_1$  and  $t_2$  is 2.

*Example 1:* In network  $A$  (Fig. 1(a)), the target multicast rate is achievable without coding, incurring link cost of 10. To reduce link cost to 9, we have to remove one of the two links between nodes  $z$  and  $w$ , making coding necessary at node  $z$ . In network  $B$  (Fig. 1(b)), by removing link  $(x, z)$ , we can establish multicast connections of rate 2 without coding using only 9 links, whereas removing one of the two links between nodes  $s$  and  $y$  first necessitates coding at node in the remaining graph. In network  $C$  (Fig. 1(c)), though one of the two links between nodes  $y$  and  $z$  is redundant, coding at node  $z$  is necessary regardless whether one of the redundant links is removed or not.  $\square$

As illustrated in Example 1, reducing link usage first by subgraph selection may give rise to necessity of coding in the remaining subgraph, but we may also choose not to do coding while allowing some extra link cost. For some networks, such as network  $A$ , minimizing link usage first always increases the requirement of coding, whereas for some others, like network  $B$ , it depends on how a minimum-cost subgraph is chosen. Also, there are networks, e.g., network  $C$ , where reducing link cost first does not increase coding cost. Hence, whether there exists such a tradeoff can be considered a topological property of a network.

If it is possible to identify the tradeoff in a given network, one may use the information to make decisions on the deployment of the network coding capability such that coding happens only at the places where significant amount of link

cost is saved, or one may not want to employ network coding at all if the amount of saved link cost turns out to be negligible for the topology. On the other hand, if the given network is known to have no such tradeoff, we can employ the aforementioned two-stage method without sacrificing optimality.

However, as will be discussed later, determining whether there exists such a tradeoff turns out to be NP-hard. In this study, we further extend our previously proposed evolutionary approaches [2], [3] to investigate the issue of the tradeoff between coding and link costs. One way to address the problem indirectly is suggested in [3], where the coding and links costs are combined to make a single objective. Such a method, however, can only provide a single solution that minimizes the combined cost. Alternatively, we may apply the method in [2] repeatedly to determine a minimal set of coding nodes/links while varying the link cost by choosing different combinations of the links to be used. This type of method, however, can be very inefficient since the possible range of link cost can be very wide.

Evolutionary algorithms, in fact, can serve more effectively as a method to identify such tradeoffs [4]. In this paper, we propose a distributed evolutionary algorithm that may reveal the utility of network coding in comparison with the amount of saved link cost, which has been unable to measure so far with any other method.

The rest of the paper is organized as follows. Section II presents the problem formulation and summarizes related work. Section III describes the problem-specific selection mechanism. Section IV presents the distributed implementation of our algorithm, while Section V analyzes the algorithm's overhead. Section VI shows experimental results and Section VII concludes with topics for future research.

## II. PROBLEM FORMULATION AND RELATED WORK

### A. Problem Formulation

We assume that the network is given by a directed multi-graph  $G = (V, E)$ , where each link has a unit capacity. Connections with larger capacities are represented by multiple links. Only integer flows are allowed, hence there is either no flow or a unit-rate 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, involving network coding or not, that enables all  $d$  sinks to receive all the information sent. We consider only 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 [5].

We assume that the given target rate  $R$  is achievable when coding is allowed at all nodes. A transmission scheme is called feasible if it achieves the target rate  $R$ . Each link  $e \in E$  is assigned link cost  $l_e$ , which is incurred when the link is used for transmission, and coding cost  $c_e$ , which is incurred if the transmission on the link involves network coding rather than simple forwarding.

Let  $f_c(x)$  and  $f_l(x)$  denote the total coding and link costs, respectively, for any transmission scheme  $x$ . We then wish to find the *Pareto optimal front*  $\mathcal{P}$  (which can be shown to be unique) defined as follows:

*Definition 1:* The Pareto optimal front is the set of the cost pairs  $(f_c^*, f_l^*)$  of a feasible transmission scheme such that there exists no other scheme  $x$  that is feasible and satisfies  $\{f_c(x) < f_c^*, f_l(x) \leq f_l^*\}$  or  $\{f_c(x) \leq f_c^*, f_l(x) < f_l^*\}$ .

*Theorem 1:* Finding the Pareto optimal front between the coding and link costs for multicast is NP-hard.

*Proof:* The problem to determine whether employing network coding at more nodes/links decreases link cost is NP-hard, because to which the NP-complete problem of computing the minimum cost for multicast without network coding [6] is reduced. To decide the converse, i.e., whether removing links increases the minimum number of coding nodes/links, is also NP-hard [3].  $\square$

### B. Evolutionary Algorithms for Network Coding

Genetic Algorithms (GAs) operate on a set of candidate solutions, called a *population*, which improves sequentially via mechanisms inspired by biological evolution [7]. Each candidate 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 above process is iterated with the newly generated population successively replacing the current one. For further details of a standard simple GA, the reader is referred to [3], [7].

For the problem of network coding resource optimization, [3] proposes a GA-based evolutionary approach, demonstrating its benefits over other existing approaches in terms of the solution quality and the applicability to a variety of generalized scenarios. Along the same direction, [2] develops a novel representation method and the associated operators, which is shown to lead to a substantial gain in the algorithm's performance, as analyzed more in depth in [8]. Furthermore, [2] presents a distributed version of the algorithm, where the resource optimization can be done on the fly integrated into a decentralized network coding framework.

### C. Evolutionary Algorithms for Multi-Objective Optimization

For optimization with multiple objectives, i.e., the coding and link costs in our case, a number of algorithms have been proposed to obtain the Pareto optimal front in a single run [4]. Commonly in those algorithms, if solution  $x$  is inferior

to another solution  $y$  with respect to one or more of the optimization criteria while the two are the same in all the remaining criteria,  $x$  is said to be *dominated* by  $y$ . More formally, in the case of the minimization problem with the two objectives  $f_c$  and  $f_l$ , the notion of domination is defined as follows:

*Definition 2:* For chromosomes  $x$  and  $y$ ,  $x$  is dominated by  $y$  (or  $y$  dominates  $x$ ) if either  $\{f_c(x) > f_c(y), f_l(x) \geq f_l(y)\}$  or  $\{f_c(x) \geq f_c(y), f_l(x) > f_l(y)\}$  holds.

Multi-objective GAs share largely the same structure with ordinary simple GAs, with some notable differences in the selection mechanism. Multi-objective selection mechanisms employ various algorithmic techniques to locate the resulting population as close to the actual Pareto optimal front as possible [4]. First, while selection in simple GAs puts more weights on the solutions with better fitness values with respect to a single objective, multi-objective GAs employ a mechanism that assigns higher probabilities for selection to less dominated solutions. In addition, multi-objective GAs employ the mechanism that preserves diversity among the solutions in an effort to obtain the full Pareto optimal front. Out of many existing multi-objective GAs, we primarily focus on Deb et al.'s NSGA-II [4], based on which we implement a novel selection mechanism specific to our problem, as will be discussed in the next section.

### III. SELECTION MECHANISM

The purpose of evolutionary algorithms in general is to serve as a stochastic search method that does not rely on any specific structures of the problem, i.e., a “black box” optimization method, so that they can be applied to as a wide variety of problems with little known structures as possible. Given a specific problem at hand, however, the desired properties of an optimization method can be very different since it is often beneficial to utilize as many structures of the given problem as possible. The insight gained from our previous application of evolutionary algorithms in the context of network coding [2], [3] is that, though the entire structure of the problem is still not known much, utilizing some problem-specific knowledge, such as the modularity among variables imposed by the given network topology, can lead to a substantial gain in the algorithm’s performance. In this section, we discuss the unique characteristics of our problem that can be incorporated into the framework of multi-objective GAs to obtain a more close-to-optimal Pareto front. Let us begin by introducing the original selection mechanism proposed by Deb et al. [4].

#### A. Original Approach

In the original NSGA-II [4], selection is done based on two criteria: non-domination rank and crowding distance. After calculating the fitness values, the algorithm finds the first *non-domination front*  $\mathcal{F}_1$ , the set of the chromosomes that are *not* dominated by any other chromosome, assigning *non-domination rank* 1 to those chromosomes. For  $i \geq 2$ ,  $i$ -th non-domination front  $\mathcal{F}_i$  consists of the chromosomes in  $P \setminus \{\mathcal{F}_1 \cup \dots \cup \mathcal{F}_{i-1}\}$  that are not dominated by others, to

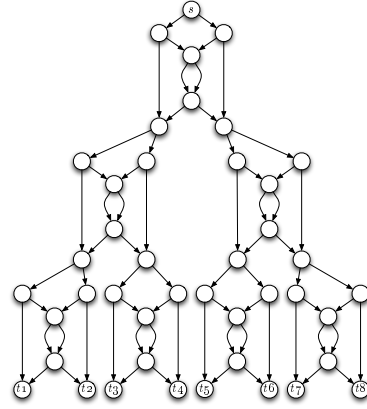


Fig. 2. Network  $G$  in Example 2

which non-domination rank  $i$  is assigned. It can be shown that each  $\mathcal{F}_i (i \geq 1)$  is nonempty unless we exhaust all the chromosomes, and thus there are only a finite number of non-domination fronts.

The chromosomes belonging to the same non-domination front are sorted with respect to each of the objectives, one after another, and the crowding distance of each chromosome is the sum of the differences between its next better and next worse chromosomes along each axis of the objectives. Intuitively, crowding distance is a measure of the densities on each non-domination front such that a chromosome in the sparse region has a high distance.

Let  $P_t$  denote the population at generation  $t$ . Then, an intermediate population  $Q_t$  of the same size  $N$  is created using a binary tournament selection, where we repeat the following procedure until  $Q_t$  is filled: a random pair of chromosomes is chosen and the one with a lower non-domination rank is selected, or if the ranks are the same, the one with a higher crowding distance is selected. After calculating the fitness values of  $Q_t$ , the  $N$  best chromosomes out of  $P_t \cup Q_t$  are selected for the actual population  $P_{t+1}$  for the next generation  $t + 1$ . At the end of the algorithm,  $\mathcal{F}_1$  of the last population is the resulting Pareto optimal front.

It is shown in [4] that non-domination rank and crowding distance can be efficiently calculated in  $O(MN^2)$  time, where  $M$  is the number of objectives and  $N$  is the population size. This NSGA-II is then evaluated to show that it surpasses other multi-objective optimization algorithms in terms of minimizing various classes of continuous-valued functions.

#### B. Problem-Specific Selection Mechanism

1) *Different Convergence Time:* Our problem displays some unique characteristics that cannot be well handled with the above selection mechanism alone. One of the hurdles in applying the above selection mechanism to our problem is that it is likely to produce only a part of the actual Pareto optimal front. More specifically, the resulting final front has a strong tendency to be skewed toward the low link cost region of the desired front.

*Example 2:* Consider the network constructed by cascading a number of copies of network  $A$  in Example 1(Fig. 1(b)) such

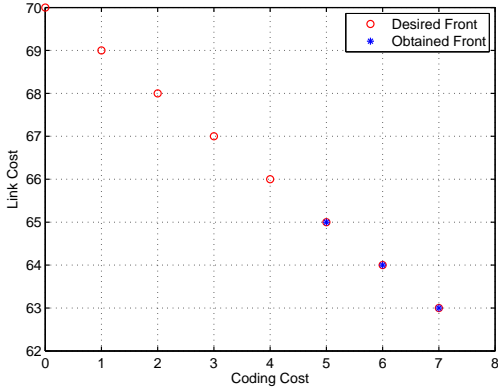


Fig. 3. Skewed Front Obtained for Network  $G$

that the source of each subsequent copy of  $A$  is replaced by an earlier copy's sink. Let network  $G$ , depicted in Fig. 2, be a depth-3 full binary tree consisting of 7 copies of  $A$ , where the source is the tree's root node and the sinks are 8 leaf nodes, and the coding and link costs of each link are both 1. For this network, the minimum number of coding links is known to be zero, but there exists a tradeoff opportunity for each copy of  $A$ . Hence, the desired Pareto front appears as the circles in Fig. 3. If we apply the evolutionary algorithm presented in [2] with the selection mechanism replaced by that in NSGA-II [4], we almost always obtain a skewed front as depicted in Fig. 3 by the stars in the lower right part of the desired full front.  $\square$

This phenomenon can be understood from the different difficulties of the optimization along the two objectives, i.e., while minimizing coding cost is NP-hard, minimizing link cost regardless of coding cost is polynomially solvable. In the context of GA, this difference translates into different convergence times to the optimal regions of the two objectives. Before the chromosomes with a low coding cost emerge, those having a low link cost are more likely to appear in earlier generations. For the same link cost, the chromosomes with a lower link cost dominate the ones with a higher link cost and thus are favored in the selection process. This selection pressure toward the low link cost region often dominates the early stage of evolution, making it hard for the population to evolve into the high link cost region which may eventually lead to low coding costs.

Therefore, we need to ensure that the chromosomes having high link costs initially, though dominated by the ones with low link costs through the middle of the iteration, are not prematurely lost, providing the algorithm with the time to find possibly the chromosomes with low coding cost and high link cost that may eventually become not dominated. To this end, we define the *coding front*  $\mathcal{C}_1$  as the set of the chromosomes having the least coding cost for each link cost level. More specifically, for each chromosome  $x \in \mathcal{C}_1$  there exists no feasible chromosome  $y$  that satisfies  $\{f_c(y) < f_c(x), f_l(y) = f_l(x)\}$ ;  $\mathcal{F}_1$  is in fact a subset of  $\mathcal{C}_1$ . We then assign non-domination rank 1 also to the chromosomes in  $\mathcal{C}_1 \setminus \mathcal{F}_1$  so that those chromosomes are not lost prematurely.

2) *Degeneracy*: Evolutionary algorithms for multi-objective optimization, as mentioned above, are typically applied to optimizing various kinds of continuous-valued function for performance evaluation [4]. In most such cases, each chromosome is directly mapped to a different value of the functions' variable(s). In our problem, however, each chromosome determines the operations performed at the interior nodes and the two discrete-valued objectives are the resulting coding and link costs, which in a sense are expressed in a parameterized form of the chromosome. Hence, there can be a substantially large number of different chromosomes that correspond to the same coding and link costs, yielding many chromosomes with zero crowding distance.

Within the original selection mechanism, the chromosomes with zero crowding distance are those least favored on the same non-domination front. However, we observed from our experiments that two chromosomes with zero crowding distance may be very different, i.e., the Hamming distance between the two may be very large. In fact, having those chromosomes that are "neutral" with respect to fitness landscape but "diverse" in terms of Hamming distance on a non-domination front may promote finding even better recombined chromosomes.

Hence, for those chromosomes with zero crowding distance, we use Hamming distance as a secondary measure of distance. More specifically, suppose that chromosomes  $x$  and  $y$ , having the same rank and zero crowding distance, compete for selection. We then compute  $h(x)$ , defined as the Hamming distance from  $x$  to the closest one among the remaining chromosomes having the identical objective values, and  $h(y)$ , defined similarly, and finally select the one with larger  $h(\cdot)$ .

#### IV. DISTRIBUTED IMPLEMENTATION

It is shown in [2] that coding resource optimization can be done in a distributed fashion integrated into a decentralized network coding framework, which is claimed as the key benefit of the approach. This characteristic carries over to the proposed evolutionary algorithm for multi-objective optimization. As can be noticed from the previous section, the changes introduced as we generalize an ordinary simple GA to a multi-objective GA involve the selection part only. Those changes translate into the modification of the data collected and distributed by the source node as well as some additional calculation at the source node, without disrupting the algorithm's main structure.

We assume that each link can transmit a fixed-size unit packet per unit time in the given direction. Each link is also assumed to be able to send some amount of feedback data, typically much smaller than the forward packet size, in the *reverse* direction. Also, we assume that each interior node operates in a burst-oriented mode; i.e., for the forward (backward) transmission, each node starts updating its output only after an updated input has been received from all incoming (outgoing) links.

The overall flow of our proposed distributed algorithm is shown in Fig. 4 with the locations of each procedure specified.

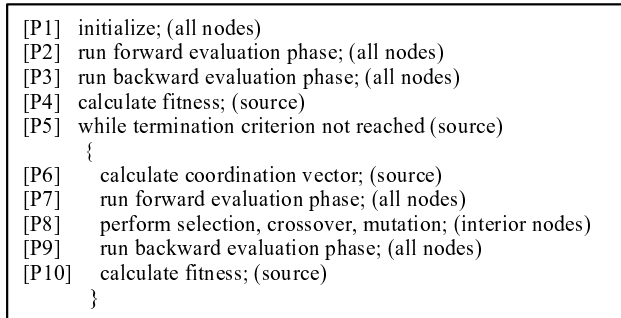


Fig. 4. Flow of Distributed Algorithm

We now proceed to describe each procedure of the algorithm in the order of the occurrence.

1) **Initialization [P1]**: The source node initiates the algorithm by transmitting the “initialize” packet containing the following predetermined parameters: the target multicast rate  $R$ , the size  $N$  of the population, the size  $q$  of the finite field to be used, crossover probability, and mutation rate. Each participating node that has received the packet forwards the packet to its downstream nodes.

Upon receiving the “initialize” packet, each node with  $d_{in}$  incoming links and  $d_{out}$  outgoing links generates a set of  $d_{out}$  random binary vectors, each of which, referred to as a *coding vector*, has length  $d_{in}$ . Note that the  $i$ -th bit of the  $j$ -th coding vector indicates whether the input from the  $i$ -th incoming link would contribute, as multiplied by a random coefficient to be determined later, to the linearly coded output on the  $j$ -th outgoing link. A set of  $d_{out}$  coding vectors is required to evaluate each chromosome, hence each node needs to manage  $Nd_{out}$  coding vectors for evaluation. However, as will be discussed later, each node must store the intermediate population  $Q$  while performing genetic operations, thus the actual size of memory required at each node is  $2Nd_{in}d_{out}$  bits.

2) **Forward Evaluation Phase [P2, P7]**: Let us first introduce the data structure used for the feasibility test of chromosomes. For each chromosome, each node transmits a *pilot vector* that consists of  $R$  components, each of which is an element of the finite field  $\mathbb{F}_q$ . The  $i$ -th component ( $1 \leq i \leq R$ ) represents the coefficient used to encode the  $i$ -th source data. We assume that a set of  $N$  pilot vectors is transmitted together by a single packet.

The source initiates the forward evaluation phase by sending out a packet containing  $N$  random pilot vectors on each of its outgoing links. Each downstream node transmits on each of its outgoing links a random linear combination of the received pilot vectors, computed based on the node’s coding vectors as follows. Let us consider a particular outgoing link and denote the associated  $d_{in}$  coding vectors by  $v_1, v_2, \dots, v_{d_{in}}$ . Let  $u_i$  ( $1 \leq i \leq N$ ) be the output pilot vector to be transmitted onto the outgoing link for evaluation the  $i$ -th chromosome. We then define the set  $J$  of indices as

$$J = \{1 \leq j \leq d_{in} \mid \text{the } i\text{-th component of } v_j \text{ is } 1\}. \quad (1)$$

For each of the  $d_{in}$  incoming links, we denote the  $i$ -th input pilot vector, out of the  $N$  pilot vectors received, by  $w_1, w_2, \dots, w_{d_{in}}$ . Then,  $u_i$  is calculated as

$$u_i = \sum_{j \in J} w_j \cdot \text{rand}(\mathbb{F}_q), \quad (2)$$

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

3) **Backward Evaluation Phase [P3, P9]**: To calculate the fitness value of each chromosome, the source node requires three kinds of information: 1) whether all the sinks can decode the data of rate  $R$ , 2) how many links are used for coding, and 3) the total cost of the links used for (either coded or uncoded) transmission.

Each sink can determine, by computing the rank of the collection of received pilot vectors, whether data of rate  $R$  is decodable for each of the  $N$  chromosomes. By inspecting its coding vectors used in the forward evaluation phase, each node can compute the number of its outgoing links where coding is needed as well as the link cost incurred at the node.

For the feedback of this information, each node transmits upstream a *fitness vector* consisting of  $N$  components, whose  $i$ -th component conveys the information needed to calculate the fitness value of the  $i$ -th chromosome. Each component of a fitness vector contains two information: the coding cost and link cost up to the location where the fitness vector is generated. An infeasible chromosome is signified by the *infinite* coding and link costs. The remaining backward evaluation phase proceeds the same way as in [2], hence we omit the details here.

4) **Fitness Calculation [P4, P10]**: The source calculates the fitness values of  $N$  chromosomes simply by summing the received fitness vectors component-wise. Note that if an infinite cost were generated by *any* of the sinks, it should dominate the summations all the way up to the source, and thus the source can detect the infeasible chromosomes.

5) **Termination Criterion [P5]**: The source node can determine when to terminate the optimization by counting the number of generations iterated thus far.

6) **Coordination Vector Calculation [P6]**: In our algorithm, we let each interior node manage the portion of the population that specifies the local operations at the node. However, overall genetic operations need to be performed in a coordinated fashion throughout the network with the following information shared: 1) which chromosomes are selected for the next generation, 2) how the selected chromosomes are paired for crossover. This information is carried by a *coordination vector* calculated at the source. The coordination vector essentially conveys the outcome of the selection mechanism described in the previous section.

We now show how the selection mechanism is implemented in our distributed setup and a coordination vector is constructed. Let us assume that, at generation  $t$ , the fitness values (not the actual chromosomes) of  $P_t$  and  $Q_t$  are available at the source, which will be shown to be valid later. From those fitness values indexed properly, the source node can

calculate the non-domination rank and crowding distance of  $P_t$  and  $Q_t$ , and then determine the indices representing the chromosomes that comprise  $P_{t+1}$ . Note that, from just those indices, each interior node can retrieve its relevant portion of  $P_{t+1}$ , if the actual chromosomes of  $P_t$  and  $Q_t$  (without the fitness values) were stored at those interior nodes. Hence, the coordination vector consists of the indices of the selected chromosomes, permuted randomly, which thus provides the paring information for crossover as well. The coordination vector is then transmitted *piggyback* onto the pilot vectors during the next forward evaluation phase, without requiring an additional procedure dedicated to it.

7) **Genetic Operations [P7]**: Based on the received coordination vector, each node can locally perform genetic operations and renew its portion of the population. For selection, each node now retains, out of  $P_t$  and  $Q_t$  saved at the node, the coding vectors that correspond to the indices contained in the received coordination vector to construct  $P_{t+1}$ .

After crossover and mutation, which are performed in exactly the same manner as in [2] (hence details are omitted), the relevant portion of  $Q_{t+1}$  is constructed at each node. The fitness values of  $Q_{t+1}$  start to be calculated as the algorithm proceeds to the forward evaluation phase of generation  $t+1$ . Note that since  $P_{t+1}$  was a subset of  $P_t \cup Q_t$ , the source node already had the fitness values of  $P_{t+1}$ , and at the time when the coordination vector for generation  $t+1$  is constructed at the source node, the fitness values of  $Q_{t+1}$  will become available at the source, which validates the assumption we made in the coordination vector calculation procedure.

## V. ALGORITHM'S OVERHEAD

### A. Complexity

For evaluation of a single chromosome, each node  $v$  computes random linear combinations of inputs in the forward evaluation phase, which requires  $O(d_{in}^v d_{out}^v R)$ . Feasibility test at each sink  $t$  is done by calculating the rank of a  $d_{in}^t \times R$  matrix, where we assume  $d_{in}^t \geq R$ , hence it requires  $O(d_{in}^t R)$ . In the backward evaluation phase, update of a fitness vector takes  $O(d_{in}^v + d_{out}^v)$ . For genetic operations, each node  $v$  requires  $O(N d_{in}^v d_{out}^v)$ . Therefore, the total computational complexity required for each generation is  $O(\sum_{v \in V} d_{in}^v d_{out}^v N R + \sum_{t \in T} d_{in}^t R)$ . Note, however, that the complexity at each node depends on just local parameters rather than the overall size of the network.

### B. Population Sizing

The size of the population often serves as an important factor for the ability of a GA to find a good solution [9]. Though it is not an easy task to predict the accurate population size required for a specific problem, it is always desirable to allow some level of flexibility in adopting a large-sized population when needed, without incurring too much complexity overhead.

In our distributed framework, the population size can be adjusted mainly by modifying the size of the packets to be used for fitness evaluation. The key observation is that the

length of a pilot vector depends on  $R$ , the desired multicast rate represented as a multiplicative factor *relative* to the unit capacity, but not the actual data rate which amounts to  $R$  times the data rate corresponding to a unit capacity. Hence, a large number (typically several hundreds) of pilot vectors can possibly be handled with a single packet transmission.

*Example 3*: To determine the size of a single pilot vector for our running example of network  $G$  considered in Example 2, we first choose the size of the finite field upon which we construct the random linear code. The limiting factor on the field size is the error probability of the randomized feasibility test, whose upper bound is given by  $1 - (1 - d/q)^\nu$ , where  $\nu$  is the maximum number of links in any set of links constituting a flow solution from the source to any receiver [10]. In  $G$ ,  $d = 8$  (number of sinks) and  $\nu = 12$ . If we desire to keep the error probability below 0.01, the smallest power  $q$  of 2 that meets the error bound is 14 and thus the length of  $N$  pilot vectors is  $N R \log_2 q = 28N$  bits. Also, the length of the fitness vector is  $N \cdot \lceil \log_2(|E| + 2) \rceil = 7N$  ( $|E| = 70$ ) bits, and the length of the coordination vector is  $N(\lceil \log_2 2N \rceil)$  bits.

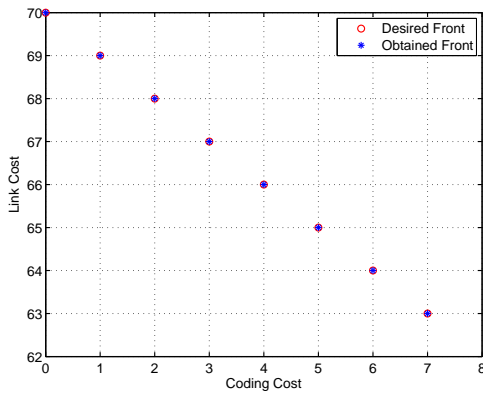
For example, if the unit packet size is 1500 bytes (the maximum Ethernet packet size), the largest  $N$  such that  $N$  pilot vectors and a coordination vector fit into a single packet turns out to be 321.  $\square$

As the population size  $N$  varies, the size of memory used at each node to store the chromosomes must also be adjusted accordingly. Also, the computational complexity required at each node during the forward and backward evaluation procedures, as well as the genetic operations, scale linearly with  $N$ . However, since each node stores only the relevant portion of the chromosomes and also the computation at each node involves only that portion of the chromosomes, the impact of increased  $N$  may be considered insignificant relative to its impact on the packet transmission.

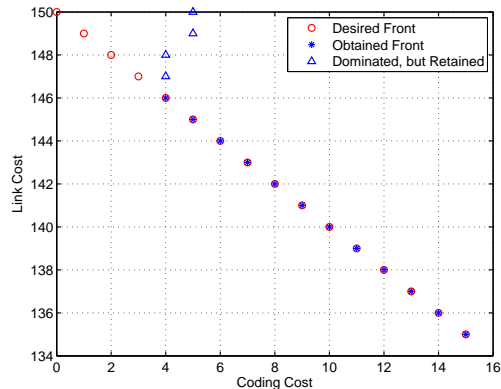
## VI. EXPERIMENTAL RESULTS

We demonstrate the performance of our algorithm by carrying out simulations on various network topologies, assuming each link has the unit coding and link costs. The parameters used for the experiments are as follows: population size  $N = 200$  and the iteration terminates after 1000 generations. Crossover and mutation rates are 0.8 and 0.02, respectively.

First, we apply our algorithm, combined with the selection mechanism described in Section III-B, to network  $G$  for which the Pareto optimal front is known. Note that, as depicted in Fig. 5(a), the algorithm now succeeds to find the whole Pareto optimal front. We also try a bigger network  $H$  of the same type, which is now a depth-4 binary tree containing 15 copies of  $A$  having 16 receivers. The resulting front is depicted by the stars in Fig. 5(b) while the triangles represent the chromosomes belonging to  $\mathcal{C}_1 \setminus \mathcal{F}_1$  and the circles are the desired Pareto optimal front. Note that the points marked by triangles represent the chromosomes that, though dominated by others, are retained in the first front. Those chromosomes may eventually converge to the desired front if they were allowed more time for evolution.



(a) Network  $G$



(b) Network  $H$

Fig. 5. Calculated Non-Domination Fronts

We also evaluate the performance of our algorithm based on the two topologies generated by the algorithm in [11] with the following parameters: (50 nodes, 87 links, 10 sinks, rate 5) and (75 nodes, 156 links, 15 sinks, rate 7), which are also experimented in [2], [8]. For the first network with 50 nodes, the obtained front is shown in Fig. 6, from which we find that it requires network coding at some links, but coding does not save the link cost. The network with 75 nodes also produces a single-point non-domination front, implying that there is no tradeoff opportunity.

## VII. CONCLUSION AND FUTURE WORK

We have proposed an algorithm, based on a multi-objective GA, that serves as a method to identify the tradeoff between the coding and link costs in multicast network coding. Our algorithm operates in a distributed manner integrated into a decentralized network coding protocol. With the proposed problem-specific selection mechanism, we have demonstrated that our algorithm effectively finds the utility of network coding in comparison with the amount of saved link cost, which has been unable to measure so far with any other method.

In the future, we may further apply the time-distributed population management scheme proposed in [12] where the population consists of a number of subpopulations updated by

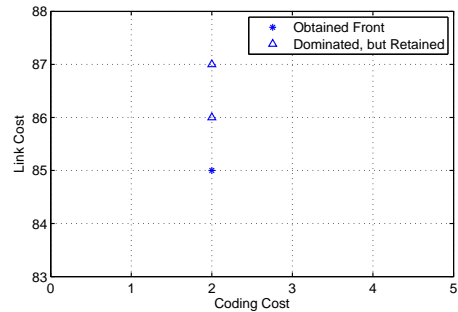


Fig. 6. Non-Domination Front Showing No Tradeoff

successive packet transmission in different time slots. Since our multi-objective GA share the overall structure with an ordinary GA, this scheme may lead to a substantial improvement on the algorithm's convergence time also in our problem. It may be also interesting to compare the performance of our algorithm, e.g., solution quality, efficiency, etc., with that of the two-stage method discussed in Section I and [2].

## REFERENCES

- [1] D. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: a decentralized approach based on network coding," in *Proc. IEEE Infocom*, 2005, pp. 1607–1617.
- [2] 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.
- [3] M. Kim, C. W. Ahn, M. Médard, and M. Effros, "On minimizing network coding resources: An evolutionary approach," in *Proc. NetCod*, 2006.
- [4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.
- [5] 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.
- [6] S. Ramanathan, "Multicast tree generation in networks with asymmetric links," *IEEE/ACM Trans. Networking*, vol. 4, no. 4, pp. 558–568, 1996.
- [7] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1996.
- [8] M. Kim, V. Aggarwal, U.-M. O'Reilly, and M. Médard, "Genetic representations for evolutionary minimization of network coding resources," in *Proc. EvoWorkshops*, 2007.
- [9] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller, "The gambler's ruin problem, genetic algorithms, and the sizing of populations," *Evolutionary Computation*, vol. 7, no. 3, pp. 231–253, 1999.
- [10] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. R. Karger, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.
- [11] G. Melançon and F. Philippe, "Generating connected acyclic digraphs uniformly at random," *Inf. Process. Lett.*, vol. 90, no. 4, pp. 209–213, 2004.
- [12] M. Kim, V. Aggarwal, U.-M. O'Reilly, and M. Médard, "A doubly distributed genetic algorithm for network coding," in *Proc. ACM Genetic and Evolutionary Computation Conference (GECCO)*, 2007.