Genetic Representations for Evolutionary Minimization of Network Coding Resources

Minkyu Kim[1], Varun Aggarwal[2], Una-May O'Reilly[2],
Muriel Médard[1], and Wonsik Kim[1]

[1] Laboratory for Information and Decision Systems
[2] Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
{minkyu@, varun_ag@, unamay@csail., medard@, wskim14@}mit.edu

Abstract. We demonstrate how a genetic algorithm solves the problem of minimizing the resources used for network coding, subject to a throughput constraint, in a multicast scenario. A genetic algorithm avoids the computational complexity that makes the problem NP-hard and, for our experiments, greatly improves on sub-optimal solutions of established methods. We compare two different genotype encodings, which tradeoff search space size with fitness landscape, as well as the associated genetic operators. Our finding favors a smaller encoding despite its fewer intermediate solutions and demonstrates the impact of the modularity enforced by genetic operators on the performance of the algorithm.

1 Introduction

Network coding is a novel technique that generalizes routing. In traditional routing, each interior network node, which is not a source or sink node, simply forwards the received data or sends out multiple copies of it. In contrast, network coding allows interior network nodes to perform arbitrary mathematical operations, e.g., summation or subtraction, to combine the data received from different links. It is well known that network throughput can be significantly increased by network coding [1, 2, 3]. While network coding is assumed to be done at all possible nodes in most of the network coding literature, it is often the case that network coding is required only at a subset of nodes to achieve the desired throughput. Consider Example 1:

Example 1. In the canonical example of network B (Fig. 1(a)) [1], where each link has unit capacity, source s can send 2 units of data simultaneously to the sinks t_1 and t_2, which is not possible with routing alone. But only node z needs to combine its two inputs while all other nodes perform routing only. If we suppose that link (z, w) in network B has capacity 2, which we represent by two parallel unit-capacity links in network B' (Fig. 1(b)), a multicast of rate 2 is possible without network coding. In network C (Fig. 1(c)), where node s is to transmit data at rate 2 to the 3 leaf nodes, network coding is required either at node a or at node b, but not at both. □

(a) Network B (b) Network B' (c) Network C
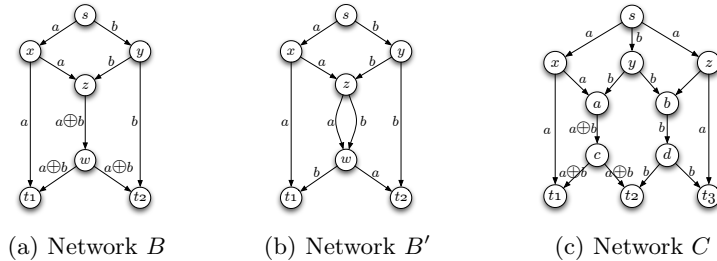
Fig. 1. Sample networks for Example 1

Example 1 leads us to the following question: To achieve the desired throughput, at which nodes does network coding need to occur? This question's answer is valuable because eliminating unnecessary coding nodes will save computation at the application layer if that is where network coding is handled. Alternatively, if network coding is integrated in the buffer management of routers, it will reduce the number of routers that need to perform coding operations without compromising communication capacity. For a GA, the problem can be posed as the minimization of coding cost (in links or nodes) subject to the constraint of feasibility (achieving the desired throughput).

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 [4]. It is shown that even approximating the minimum number of coding nodes within any multiplicative factor or within an additive factor of $|V|^{1-\epsilon}$ is NP-hard [5]. Note, however, that once the set of coding nodes is identified, a network code achieving the desired throughput can be efficiently constructed for the multicast scenario, either in a deterministic [6] or randomized fashion [7].

In the network research community, [8] and [9] have documented results that demonstrate the benefit of the GA over other existing approaches in terms of reducing the number of coding links or nodes and its applicability to a variety of generalized scenarios. These contributions emphasized the computational "how-to" aspects of feasibility checking and the transformations of the network graph into secondary graphs that express possible coding situations uniformly since these are key to evaluating the fitness function of the GA.

In the course of investigating the feasibility checks, graph transformations, and value of using a GA, we experimented with two different genotype encodings[1] and associated operators. For both encodings, we use a genotype composed of a number of blocks, each of which consists of a set of variables indicating the link states. For a block of length k, using an alphabet of cardinality 2, the *Binary Link State* (BLS) encoding represents all possible 2^k states of k links. On the other hand, for the *Block Transmission State* (BTS) encoding, we group

---

[1] To minimize confusion, throughout the paper, the term "encoding" refers to "genotype encoding" only, while the term "coding" means "network coding."

those link states into $(k + 2)$ transmission states. Despite the smaller search space size of BTS encoding, it is not clear that it should be superior to BLS encoding because in grouping many link states into one, less information that would relate the fitness of solutions *intermediate* to the best solution is available in contrast to BLS encoding which provides more information through its intermediate solutions.

In this paper we focus on the two different encodings with associated genetic operators and conduct a more comprehensive comparison between them. Specifically relevant to the GA community, we consider into the GA encoding tradeoff issues related to search space size and fitness landscape. The rest of the paper is organized as follows. Section 2 presents the problem formulation, and Section 3 describes the network coding GA (NCGA) with the two different encodings and associated operators. Section 4 sets up a set of experiments into relative values of the encodings and discusses the results. Section 5 presents a summary of the results and our conclusions.

2   Problem Formulation

We assume that a network is given by a directed multigraph $G = (V, E)$ as in [10] where each link has a unit capacity whose unit can be arbitrarily chosen, e.g., k bits per second for a constant k, or a fixed size packet per unit time, etc. 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. 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. Rate R is said to be achievable if there exists a transmission scheme that enables all $|T|$ sinks to receive all of the information sent. 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 [2].

Given an achievable rate R, we wish to determine a minimal set of nodes where coding is required in order to achieve this rate. However, whether coding is necessary at a node is determined by whether coding is necessary at at least one of the node's outgoing links and thus, as pointed out also in [5], the number of coding links is in fact a more accurate estimator of the amount of computation incurred by coding. We assume hereafter that our objective is to minimize the number of coding *links* rather than *nodes*. Note, however, that as demonstrated in [8], it is straightforward to generalize the proposed algorithm to the case of minimizing the number of coding nodes. Furthermore, [8] shows that, with appropriate changes, the algorithm can be readily applied to more generalized optimization scenarios, e.g., when links and nodes have different coding costs.

It is clear that no coding is required at a node with only a single input since these nodes have nothing to combine with [8]. For a node with multiple incoming links, which we refer to as a *merging node*, if the linearly coded output to a particular outgoing link weights all but one incoming message by zero, 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 [5]. Thus, to determine whether coding is necessary at an outgoing link of a merging node, we need to verify whether we can constrain the output of the link to depend on a single input without destroying the achievability of the given rate. As in network C of Example 1, the necessity of coding at a link depends on which other links code and thus the problem of deciding where to perform network coding in general involves a selection out of exponentially many possible choices. We employ a GA-based search method to efficiently address the large and exponentially scaling size of the space.

3   Network Coding GA (NCGA)

Prior to using the NCGA, the given network graph G is transformed into a secondary graph by either of the two methods presented in $[8, 9]^2$. Regardless which method is used, mapping the network coding problem to a GA framework is done as follows.

Suppose a merging node with $k(\geq 2)$ incoming links. To consider the transmission to *each* of its outgoing links, we assign a binary variable to each of its k incoming links, which being 1 indicates that the link state is *active* (the input from the associated incoming link is transmitted to the outgoing link) and 0 indicates it is *inactive*. Given that network coding is required for the transmission only if two or more link states are active, we may need to consider those k variables together. We refer to the set of the k variables as a *block* of length k (see Fig. 2 for an example). The way how those binary variables are actually encoded as a genotype will be described later in this section.

(a) Merging node v (b) Two blocks for outgoing links of v
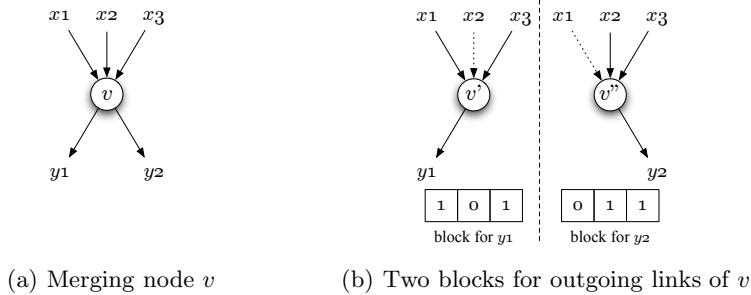
Fig. 2. Node v with 3 incoming and 2 outgoing links results in 2 blocks, each with 3 variables indicating the states of incoming links (x_1, x_2, x_3) onto the associated outgoing link

Constraint and Fitness Function. A genotype is called *feasible* if there exists a network coding scheme that achieves the given rate R with the link states

---

^{2} An interested reader is referred to [8,9] for the details of the two methods for graph transformation and feasibility testing.

determined by the genotype. To calculate the fitness of genotype $\underline{y}$, its feasibility must be checked by either of the two methods in [8], [9] depending on the secondary graph chosen earlier, and the fitness value F is assigned as

$$F(\underline{y}) = \begin{cases} \text{number of blocks with two or more active links,} & \text{if } \underline{y} \text{ is feasible,} \\ \infty, & \text{if } \underline{y} \text{ is infeasible.} \end{cases}$$

The NCGA uses a standard generation-based GA control loop with tournament selection. It terminates at some maximum number of generations. Afterward, the best solution of the run is optimized with *greedy sweep*: we switch each of the remaining 1's to 0 if it can be done without violating feasibility. This procedure may only improve the solution, and sometimes the improvement can be substantial. Reference [9] proves that the NCGA with greedy sweep is guaranteed to perform no worse than the existing algorithm in [5].

Binary Link State (BLS) Encoding and Operators. This encoding allows a block of length k to take any of 2^k possible binary strings of length k. If we denote by d_{in}^v and d_{out}^v the in-degree and the out-degree of node v, node v has d_{out}^v blocks of length d_{in}^v, and we have a total of $m = \sum_{v \in \mathcal{V}} d_{in}^v d_{out}^v$ binary variables, where $\mathcal{V}$ is the set of all merging nodes. We must explore the m-dimensional binary space of 2^m candidates to find the desired minimal set of coding links.

For BLS encoding we use uniform crossover, where each pair of genotypes is selected for crossover with a given probability (mixing ratio) and the two genotypes in a selected pair exchange each bit independently with another given probability (crossover probability). For mutation, we use simple binary mutation, where each bit in each genotype is flipped independently with a given probability (mutation rate). Since these operators deal with each bit separately, we refer to the operators used for BLS encoding as *bit-wise* genetic operators.

Block Transmission State (BTS) Encoding and Operators. As mentioned above, once a block has at least two 1's, replacing all the remaining 0's with 1's has no effect on whether coding is done. Moreover, it can be shown that substituting 0 with 1, as opposed to substituting 1 with 0, does not hurt the feasibility. Therefore, for a feasible genotype, any block with two or more 1's can be treated the same as the block with all 1's. Thus we could group all the states with two or more active links into a single state, *coded* transmission. This state is rounded out by k states for the *uncoded* transmissions of the input received from one of the k single incoming links and one state indicating *no* transmission. Thus BTS encoding emerges where each block of length k can only take one of the following $(k + 2)$ strings: "111...1", "100...0", "010...0", "001...0", ..., "000...1", "000...0". The net effect is a reduction in the number of possible states for a block to $(k + 2)$ rather than 2^k. If we let w be the total number of blocks (i.e., $w = \sum_{v \in \mathcal{V}} d_{out}^v$) and k_i denote the length of the i-th block ($i = 1, ..., w$), the search space size is $\prod_{i=1}^{w} (k_i + 2)$. However, the benefit of the smaller space size in fact comes at the price of losing the information on the partially active link states that may serve as intermediate steps toward an uncoded transmission state. This tradeoff will be discussed more in depth in Section 4.

To preserve the BTS encoding structure throughout genetic operations, we need to define a new set of genetic operators, which we refer to as *block-wise* genetic operators. For block-wise uniform crossover, we let two genotypes subject to crossover exchange each block, rather than bit, independently with the given crossover probability. For block-wise mutation, we let each block under mutation take another string chosen uniformly at random out of $(k + 1)$ other strings for a length-k block. If mutation rate is α, the average number of changed bits in a length-k block is now calculated as $\frac{4k^2}{(k+1)(k+2)}\alpha$, whereas it is $k\alpha$ for bit-wise mutation. While the difference becomes more apparent when k is large since the latter is upper bounded by 4α, those values are still different for $k = 2$, where the block structure makes no difference in the space size. Though block-wise mutation may lead to much smaller number of flipped bits, it is more likely to cause a sudden change in a genotype.

4   Experiments

4.1   Experiment Setup

The two encodings not only differ in the size of search space, but in the way the genetic operators are applied. In BTS encoding with the block-wise operators, crossover is applied at the block boundaries and mutation is performed intra-block. However, in BLS encoding with the bit-wise operators, crossover and mutation are randomly applied without respecting any block boundaries. While evaluating the effect of the search space size reduction, we also want to investigate whether the exploitation of block level modularity by the block-wise operators gives any significant improvement in the algorithm's performance. We thus set up two experiments: Experiment I compares the effect of the two encodings combined with associated operators on the performance of the NCGA, while Experiment II tests the effect of the operators alone by isolating the effect of the encodings that lead to different space sizes.

Experiment I: We use two acyclic networks, I-50 and I-75, generated by the algorithm in [11], whose details are given in Table 1. Note that BTS encoding reduces the size of the search space by 30.3 and 115 orders of magnitude for networks I-50 and I-75, respectively, compared with that in the case of BLS encoding. This experiment tests which encoding is better given the tradeoff in the search space size and ease of traversing the fitness landscape.

Experiment II: We construct a set of synthetic networks with only blocks of length 2. Note that for a block of length 2, the two encodings have the same search space size ($2^k = k + 2$ when $k = 2$), but the block-wise operators retain their modularity. These networks are constructed by cascading a number of copies of network B' in Fig. 1(b) such that the source of each subsequent copy of B' is replaced by an earlier copy's sink. We use fixed-depth binary trees containing 3, 7, 15, and 31 copies of B' (henceforth called II-3, II-7, II-15, and II-31, respectively). Parameters of these networks are given in Table 1. All these

network have 0 as the minimum number of coding links, i.e., multicast rate 2 is achievable without coding. We scale up the network size to investigate the payoff one gets with modular operators as the search space size increases.

We use the NCGA with the decomposition-based graph transformation and the max-flow feasibility testing described in [9]. For comparison, we also perform experiments using the two existing approaches by Fragouli et al. [12] ("Minimal 1")[3], and Langberg et al. [5] ("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.

Table 1. Details of the networks used in the experiments

Network	Genotype length	Number of blocks	Avg. length of blocks	Search space size ($\log_{10}$) (BLS/BTS)
I-50	280	71	3.94	84.29/53.93
I-75	761	130	5.85	229.08/113.47
II-3	32	16	2	9.63/9.63
II-7	80	40	2	24.08/24.08
II-15	176	88	2	52.98/52.98
II-31	368	184	2	110.78/110.78

4.2 Algorithm Parameters

We set the total budget of fitness evaluations to 150,000 (a very small fraction of the search space size of the networks considered. Preliminary experiments suggested different tournament sizes and mutation rates for the two encodings: 10 and 0.006 for BLS encoding, and 100 and 0.012 for BTS encoding, respectively. All other parameters are matched for the two encodings. We perform 30 runs for each network with both encodings. Table 2 summarizes the parameters used.

Table 2. GA parameters used in the experiments

Population size	150
Tournament size	10(BLS)/100(BTS)
Maximum generations	1000
Mixing ratio/Crossover probability	0.8/0.8
Mutation rate	0.006(BLS)/0.012(BTS)

When randomly initializing the population, we insert an all-one vector, which represents the solution where coding is done everywhere and thus is feasible by the assumption that the given rate R is achievable. The role of the all-one vector

---

[3] Though minimizing network coding resources is not its main concern, [12] presents an algorithm to obtain a subgraph with a minimal number of coding links.

as a feasible starting point is crucial to the performance of the algorithm as discussed in [8].

4.3   Experimental Results

Results for the both experiments are summarized in Table 3. The table shows the optimal fitness achieved by each algorithm averaged over 30 runs. The statistical significance of the difference between BTS and BLS encodings is measured by conducting paired t-tests and the p-values are reported in the last row of the table.

Table 3. Performance of the algorithms for each network. Each value in brackets is standard deviation.

	I-50	I-75	II-3	II-7	II-15	II-31
NCGA/BLS	3.33(1.03)	6.43(1.30)	0.93(0.69)	2.20(1.27)	5.57(1.55)	12.43(2.37)
(w/o greedy sweep)	3.33(1.03)	39.93(2.74)	0.93(0.69)	2.20(1.27)	5.57(1.55)	12.43(2.37)
NCGA/BTS	2.40(0.62)	3.63(0.61)	0.00(0.00)	0.00(0.00)	0.17(0.38)	1.03(0.81)
(w/o greedy sweep)	2.40(0.62)	3.63(0.61)	0.00(0.00)	0.00(0.00)	0.17(0.38)	1.07(0.83)
Minimal 1	4.90(1.37)	9.50(2.16)	3.00(0.00)	7.00(0.00)	15.50(0.00)	31.00(0.00)
Minimal 2	4.33(1.37)	7.90(1.71)	2.13(0.86)	4.37(1.25)	9.90(1.65)	19.97(2.66)
p-value	$8.21e^{-5}$	$2.65e^{-15}$	$6.7e^{-10}$	$2.2e^{-13}$	$2.9e^{-26}$	$1.55e^{-32}$

Experiment I: For both networks I-50 and I-75, the NCGA with greedy sweep, with either of the two encodings, outperforms the two existing minimal approaches. Between the two encodings, BTS encoding gives rise to a substantial performance gain over BLS encoding with the statistical significance confirmed by the tabulated p-values.

Experiment II: Again the NCGA with BTS encoding outperforms that with BLS encoding on average for all networks, while either of the two performs significantly better than the minimal algorithms. For networks II-3 and II-7, the NCGA with BTS encoding finds the optimal (0 coding links) in all of the 30 runs. For networks II-15 and II-31, it succeeds to find the optimal solution 25 and 8 times, respectively. On the other hand, BLS encoding does not find the optimal number of coding links in any of the 30 runs for networks II-7, II-15, and II-31.

The average performance of the NCGA with both encodings is plotted against the logarithm of the search space size in Fig. 3. The plot suggests a linear scaling of algorithms as the search space size grows exponentially. More data points would lend more confidence to this hypothesis. The curve for BTS encoding has a much smaller intercept and slope than BLS encoding, implying that the payoff of the block-wise operators increases as the search space size increases.

4.4   Discussion of Results

Experiment I clearly indicates that BTS encoding is better than BLS encoding for the networks considered. We can thus conclude that the benefits of the smaller
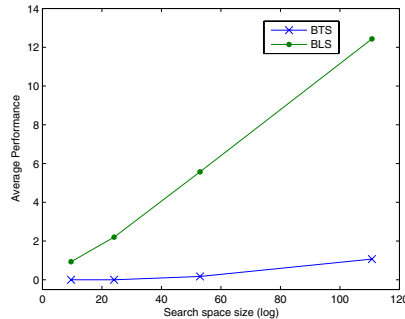
Fig. 3. Average performance of NCGAs with log of search space size

search space trump the challenge of the more difficult fitness landscape. For network I-50, BTS encoding improves over BLS encoding on average by a single coding link. Though small, this difference is statistically significant. For network I-75, without greedy sweep, the average difference in performance between the two algorithms is much higher, i.e., 34 coding links. This large difference in performance can be attributed to two specific factors: the much larger search space size (see Table 1) and larger average block size. The difference also indicates that the information on the intermediate solutions that BLS encoding provides may not be particularly useful without guaranteeing that those intermediate steps ultimately lead to an uncoded transmission state.

Experiment II demonstrates the superiority, by a remarkably large margin, of the block-wise operators over the bit-wise operators. It also indicates that both NCGAs scale linearly with an exponentially growing search space size (see Fig. 3), which is remarkable. This prompts due analysis of the difference between the two operators. When applied to the pair of blocks "00" and "11", the block-wise crossover cannot result in either block "01" or "10". However, for the bit-wise crossover, the pair of blocks "00" and "11" may result in "00", "01", "10", or "11". It can be shown that with probability $\frac{1}{4}$ the two crossovers behave differently, if the population has equal frequency of all block types. Let us recall that the block-wise mutation leads to a smaller number of changed bits on average than the bit-wise mutation (Section 3). Nevertheless, the block-wise mutation exhibits higher "exploratory power" than the bit-wise mutation in the sense that it is more likely to lead to changes in multiple bits. For the block-wise mutation, given any block, the remaining three blocks are equally likely to occur on mutation. Thus, if mutation rate is α, the probabilities of 0, 1, 2-bit change are $1 - \alpha$, $\frac{2}{3}\alpha$, $\frac{1}{3}\alpha$, respectively, whereas those probabilities in the bit-wise case are $(1 - \alpha)^2$, $2\alpha(1 - \alpha)$, α^2, respectively. Provided that $\alpha < \frac{1}{3}$, the probability of 2-bit change is larger for the block-wise mutation. A similar analysis can be done for the whole genotype as well.

One may speculate that the better performance of the block-wise operators is due to the higher exploratory power of the block-wise mutation rather than the modularity of the operators. To confirm the contrary, we consider a new set of

Table 4. Performance of NCGA with MHD operators. Refer to Table 3 for comparison with NCGAs with bit-wise or block-wise operators.

	II-3	II-7	II-15	II-31
NCGA/MHD	0.77(0.68)	2.47(1.33)	5.83(1.68)	12.63(3.23)

operators, called the *Matched Hamming Distance* (MHD) operators, where the MHD mutation leads to the statistically same Hamming distance changes as the block-wise mutation, but exhibits no positional bias as to where the mutation is applied, and the MHD crossover is the same as the bit-wise crossover which neither imposes modularity. From Table 4 compared with Table 3, we observe that the MHD operators perform similarly as the bit-wise operators, but far worse than the block-wise operators. We can thus confidently claim that the respect for modularity enforced by the block-wise operators is the main cause of the superior performance of the block-wise operators.

5   Conclusions and Future Work

For our suite of network coding problems, we have found that the benefits of the smaller search space and modular operators trump the challenge of the more difficult fitness landscape. In the future, we will study the effect of exploiting further modularity with BTS operators that cross over at merging node boundaries and perform intra-block mutations. We could incorporate hierarchical modularity using domain knowledge of sparsely connected regions, regions with similar structure or simply neighboring nodes. A hierarchical structure of crossover boundaries could be formed and applied with different probabilities. These results will inform the GA community and help push the state-of-art in algorithms for minimizing network coding resources.

References

1. Ahlswede, R., Cai, N., Li, S.Y.R., Yeung, R.W.: Network information flow. IEEE Trans. Inform. Theory **46**(4) (2000) 1204–1216
2. Li, S.Y.R., Yeung, R.W., Cai, N.: Linear network coding. IEEE Trans. Inform. Theory **49**(2) (2003) 371–381
3. Fragouli, C., Le Boudec, J.Y., Widmer, J.: Network coding: An instant primer. SIGCOMM Comput. Commun. Rev. **36**(1) (2006) 63–68
4. Richey, M.B., Parker, R.G.: On multiple Steiner subgraph problems. Networks **16**(4) (1986) 423–438
5. Langberg, M., Sprintson, A., Bruck, J.: The encoding complexity of network coding. In: Proc. IEEE ISIT. (2005)
6. Jaggi, S., Sanders, P., Chou, P.A., Effros, M., Egner, S., Jain, K., Tolhuizen, L.: Polynomial time algorithms for multicast network code construction. IEEE Trans. Inform. Theory **51**(6) (2005) 1973–1982
7. Ho, T., Koetter, R., Médard, M., Karger, D.R., Effros, M.: The benefits of coding over routing in a randomized setting. In: Proc. IEEE ISIT. (2003)

8. Kim, M., Ahn, C.W., Médard, M., Effros, M.: On minimizing network coding resources: An evolutionary approach. In: Proc. NetCod. (2006)

9. Kim, M., Médard, M., Aggarwal, V., O'Reilly, U.M., Kim, W., Ahn, C.W., Effros, M.: Evolutionary approaches to minimizing network coding resources. In: Proc. IEEE Infocom (to appear). (2007)

10. Koetter, R., Médard, M.: An algebraic approach to network coding. IEEE/ACM Trans. Networking **11**(5) (2003) 782–795

11. Melançon, G., Philippe, F.: Generating connected acyclic digraphs uniformly at random. Inf. Process. Lett. **90**(4) (2004) 209–213

12. Fragouli, C., Soljanin, E.: Information flow decomposition for network coding. IEEE Trans. Inform. Theory **52**(3) (2006) 829–848