Efficient Resource Minimization Scheme for Network Coding-Assisted Multicast System

Seyed Amin Hejazi, M. Naeem and D. C. Lee[1], Senior *Member, IEEE*

Abstract— **In this paper, we consider the problem of minimizing the resources used for network coding (MRUNC) while achieving the desired throughput in a multicast system. The problem of minimizing the number of network coding links is NP-hard. In this paper we propose a low- complexity Estimation of Distribution Algorithm (EDA) for MRUNC. Our EDA is applicable to the network with and without cycles. The numerical results show the effectiveness of the proposed method over previously proposed algorithms.**

***Index Terms:*— Estimation of Distribution Algorithm (EDA), Network Coding, Max-Flow.**

I. INTRODUCTION

THE main challenge in communication networks can be summarized as increasing the throughput of information transmission while minimizing the cost of the system. The network topology and link capacities limit the throughput of information flow within a data network. Recently, network coding [1][2] has been receiving much attention as an interesting way of increasing the throughput. Furthermore, network coding has been successfully applied to computer networks [3], wireless sensor networks [3], and peer-to-peer systems [4].

The basic idea of network coding is well presented in the seminal paper by Ahlswede, et. al. [1]. It has been shown that the throughput of the information flow from the source to sinks in the multicast system can be improved by intermediate nodes' activity of encoding the outgoing messages on the basis of the incoming messages. In [5], Li et al. further showed that linear network coding is sufficient to achieve the optimal throughput in the multicast system.

A classic example that illustrates the power of network coding is shown in Fig. 1, where each link has unit capacity. The achievable throughput in multicasting from source *s* to sinks d1 and d2 in the network A of Fig. 1, with network coding, is 2. Fig. 1 illustrates that node *a* in network A encodes the message that is transmitted from node *a* to node *b* through linear operation $\alpha+\beta$. Fig. 1 also illustrates that sink d1 can obtain both messages α and β by performing linear operations from messages α and $\alpha+\beta$, which come through links (*m, d1*) and (*b, d1*), respectively.

[1] The authors are with the School of Engineering Science at Simon Fraser University, Burnaby, BC V5A 1S6 Canada. (e-mail:shejazi@sfu.ca; mnaeem@sfu.ca; dchlee@sfu.ca).

This work was supported in part by a Discovery Grant of the Natural Science and Engineering Research Council (NSERC) of Canada.

Similarly, sink *d2* can obtain both messages α and β by performing linear operations from message β, which comes through link *(n, d2)* and message $\alpha+\beta$, which comes through link *(b, d2)*. Each of nodes d1 and d2 can obtain both messages α and β, so the throughput can be 2 per unit message transmission time. It turns out that network A cannot achieve multicast throughput 2 without network coding. It has been proven in [1] that multicast throughput of *h* can be achieved from source *s* to sinks *d1*, *d2*… *dl* if

$$h = \min_l \ \max\_\text{flow}(s, dl), \tag{1}$$

where $\max\_\text{flow}(s, dl)$ denotes the maximum flow from node *s* to *dl* in the graph. However, this statement does not prescribe which links should employ network coding, like link *(a,b)* in Fig. 1. For the purpose of reducing cost and complexity, it is desired to achieve a specified multicast throughput with as small number of coding links as possible. It is computationally complex to determine a minimal set of nodes where coding is required. It can be shown that the problem of minimizing the number of coding links to achieve a specified thorughput is *NP-hard* [6].

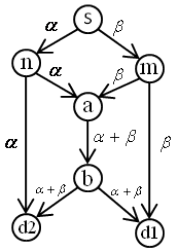

Fig. 1. Sample Networks

There are heuristic algorithms that try to find a small set of coding links that achieve a specified multicast rate toward the goal of minimizing the number of coding links [7][8][9][10]. Among these, the genetic algorithms presented by Kim *et al.* in [9] [10] appear to perform the best in terms of the number of coding links. Genetic algorithms are heuristic in nature, so they find a suboptimal solution.

In the present paper, we propose another evolutionary algorithm, Estimation-of-Distribution algorithm (EDA), for finding a small set of coding links that achieve the specified multicast throughput. EDA creates a new population from the probability distribution estimated from previous generations instead of string manipulation [13] of GA. No significant parameter tuning is required for EDA as compared to other Evolutionary Algorithms (EAs). Our experiments with EDA show that the number of coding links is significantly reduced compared to the aforementioned methods. Moreover, the optimal solution is derived much faster and within a smaller number of iterations.

II. Problem Formulation

As in [9-10], we model the network by a directed multi-graph $G = (V, E)$, where each link has a unit capacity and connections with larger capacities are represented by multiple unit capacity links [2]. Only integer flows are allowed, so for our network, there is either no flow or a unit rate of flow on each link. This formulation enables the graph decomposition method [10] to facilitate the optimization formulation. Modeling a network by a direct multi-graph with unit-capacity links is not too restrictive if the link capacities are discrete. We consider a single source multicast scenario in which a single source $s \in V$ wishes to transmit data at given rate R to a set $T \subset V$ of receiver nodes (sinks), where $|T| = d$. The rate R is said to be achievable if there exists a network coding scheme that enables all d sinks to receive the information correctly. Our optimization problem is to find a minimal set of coding links that achieve multicast rate R. We only consider linear coding because linear coding is sufficient for minimizing the number of coding links to achieve a specified multicast throughput [5]. For network coding, we do not need to consider outgoing links from a node that has less than two incoming links. Such outgoing links cannot perform a linear operation on different messages coming through different incoming links. For coding links, we only need to consider outgoing links from the nodes with multiple incoming links. We refer to a node with multiple incoming links as a *merging node*.

Now, we define our optimization variables and the objective function. For each merging node with $d_{in} > 1$ incoming links and $d_{out} > 0$ outgoing links, we define 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. $a_{ij} = 1$ indicates that the messages from incoming link i contribute to the linearly coded output message forwarded through outgoing link j. $a_{ij} = 0$ indicates that the messages from incoming link i are not considered in encoding a message to be forwarded through outgoing link j. For the jth outgoing link, we refer to the set of associated binary variables $a_j = (a_{ij})_{i \in \{1,...,d_{in}\}}$ (block) as a coding vector (Fig. 2). Network coding is required over link j if $\|a_j\|_1 \equiv \sum_{i \in \{1,...,d_{in}\}} a_{ij} \geq 2$. Our optimization problem is:

$$\min \sum_{J \in \tilde{E}} q_J(a_J) \qquad (2)$$

SUBJECT TO: transmit data at given rate R to d receiver nodes

where

$$q_J(a_J) = \begin{cases} 1, & \text{if } \|a_J\|_1 \geq 2 \\ 0, & \text{otherwise} \end{cases}$$

and $\tilde{E}$ denotes the set of links that are outgoing from a merging node.

III. A Brief Introduction to EDA

Evolutionary algorithms have been often used to solve difficult optimization problems. Candidate solutions to an optimization problem are represented as individuals in the population. In EAs the cost function value of a candidate solution to the optimization problem indicates the fitness of the individual in the concept of natural selection [11]. Unlike other evolutionary algorithms, in EDA a new population of individuals in each iteration is generated without crossover and mutation operators. Instead, in EDA a new population is generated based on a probability distribution, which is estimated from the best selected individuals of previous iteration [12].

In general, conventional EDAs can be characterized by parameters and notations (I_s, F, Δ_l, η_l, β_l, p_s, Γ, I_{Ter})[13] where

1. I_s is the space of all potential solutions (entire search space of individuals).
2. F denotes a fitness function.
3. Δ_l is the set of individuals (population) at the l_{th} iteration.
4. η_l is the set of best candidate solutions selected from set Δ_l at the l_{th} iteration.
5. We denote $\beta_l \equiv \Delta_l - \eta_l \equiv \Delta_l \cap \eta_l^c$ .where η_l^c is the complement of η_l.
6. p_s is the selection probability. The EDA algorithm selects $p_s|\Delta_l|$ individuals from set Δ_l to make up set η_l.
7. We denote by Γ the distribution estimated from η_l (the set of selected candidate solutions) at each iteration
8. I_{Ter} are the maximum number of iteration

In conventional EDAs, each individual is designated by a binary string of length n (n-dimensional binary vector). We denote $X = (x_1, x_2, \cdots, x_n)$ as an individual by a binary row vector, $x_i \in \{0,1\}$. An EDA maintains a population of individuals in each iteration. We denote the number of individuals in population by $|\Delta_l|$. Population Δ_l can be specified by the following matrix:

$$X = \begin{pmatrix} X^1 \\ X^2 \\ \vdots \\ X^{|\Delta_l|} \end{pmatrix} = \begin{pmatrix} x_1^1 & x_2^1 & \vdots & x_n^1 \\ x_1^2 & x_2^2 & \vdots & x_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ x_1^{|\Delta_l|} & x_2^{|\Delta_l|} & \vdots & x_n^{|\Delta_l|} \end{pmatrix} \qquad (3)$$

where superscript j in the row vector $X^j = (x_1^j, x_2^j, x_3^j, ..., x_n^j)$ indexes an individual in the population. A typical EDA is described in the following steps:

Step 0: *Generate initial population Δ_0. The initial population ($|\Delta_0|$ individuals) is typically obtained by sampling according the uniform (equally likely) distribution [12]:*

$$p(\theta_1, \theta_2, \cdots, \theta_n) = \prod_{i=1}^{n} p_i(\theta_i), \qquad (4)^2$$

$$\forall i = 1, 2, ..., n, \text{ and } p_i(\theta_i = 1) = p_i(\theta_i = 0) = 0.5$$

For iterations l = 1,2, ..., follow steps 1 through 6

Step 1: *Evaluate the individuals in the current population Δ_{l-1} according to the fitness function F. Sort the candidate solutions (individuals in the current population) according to their fitness orders.*

Step 2: *If the best candidate solution satisfies the convergence criterion or the number of iterations exceeds its limit I_{Ter} , then terminate; else go to step 3.*

[2] In conventional EDAs, the joint probability distribution from which the initial population is sampled is a product of 'n' univariate marginal probability distributions, each following a Bernoulli distribution with parameter value equal to 0.5. Later in this paper, we present different methods of generating the initial population).

Step 3*: Select the best* $p_s\Delta_{l-1}$ *candidate solutions (individuals) from current population* Δ_{l-1}*. This selection is accomplished according to the sorted candidate solutions.*

Step 4*: Estimate the probability distribution* $p(\theta_1,\theta_2,\cdots,\theta_n)$ *on the basis of* $|\eta_{l-1}|$ *best candidate solutions. We denote this estimation by*

$$\Gamma = P(\theta_1,\theta_2,\cdots,\theta_n\,|\eta_{l-1}) \tag{5}$$

Step 5*: Generate new* $|\Delta_{l-1}|-|\eta_{l-1}|$ *individuals on the basis of this new estimated probability distribution* Γ*. Replace the bad* $|\beta_{l-1}|$ *individuals with newly generated* $|\Delta_{l-1}|-|\eta_{l-1}|$ *individuals.*

Step 6*: Go to step 1 and repeat the steps*

We followed the steps of the above pseudo code for our EDA implementation problem. In our experimentation, for estimation (5), we used a simple scheme of estimating the marginal distributions separately and using product form

$$\begin{aligned}\Gamma = p(\theta_1,\theta_2,\cdots,\theta_n\,|\eta_{l-1}) &= \prod_{i=1}^{n} p_i(\theta_i\,|\eta_{l-1}) \\ &= \prod_{i=1}^{n}\left(\frac{\sum_{j=1}^{|\eta_{l-1}|}\delta\left(x_i^j=\theta_i\,|\eta_{l-1}\right)}{|\eta_{l-1}|}\right)\end{aligned} \tag{6}$$

where δ is an indicator function and it can be expressed as

$$\delta\left(x_i^j=\theta\,|\eta_{l-1}\right)=\begin{cases}1 & \text{if } x_i^j=\theta\\ 0 & \text{otherwise}\end{cases} \tag{7}$$

Even with this simple application of EDA, the simulation results show that the performance of EDA is better than previously proposed algorithms. In addition, we were able to modify this basic EDA algorithm and even further improve the algorithm's performance.

IV. Components of EDA for our Problem

A. *Individual Representation*

Each individual (a candidate solution) is represented by acollection of binary vectors (strings) $\left\{a_J\,\middle|J\in\tilde{E}\right\}$. Forexample, if we have just two merging nodes *v1* and *v2* in the network, we can represent an individual as shown in Fig. 2. Vector (or binary string) a_J indicates whether outgoing link *J* is allowed to code.

B. *Fitness Function and Constraint Check*

It is natural to use the objective function in optimization problem (2) as the fitness function of our EDA. If a candidate solution (an individual in EDA) does not satisfy the constraint in (2), we disregard that candidate as a valid solution and replace it with a feasible candidate, as will be explained in the following sections. We now discuss how to check if a candidate solution $\left\{a_J\,\middle|J\in\tilde{E}\right\}$ satisfies the constraint.

Note that the maximum achievable multicast rate is (1) if all links are allowed to be a coding link. However, this statement does not say what the achievable multicast rate is if we prevent a particular set of links from being a coding link. A candidate solution $\left\{a_J\,\middle|J\in\tilde{E}\right\}$ precludes every link *J* with property $\|a_J\|_1<2$ from coding. In order to examine the achievability of a specified multicast rate for a particular

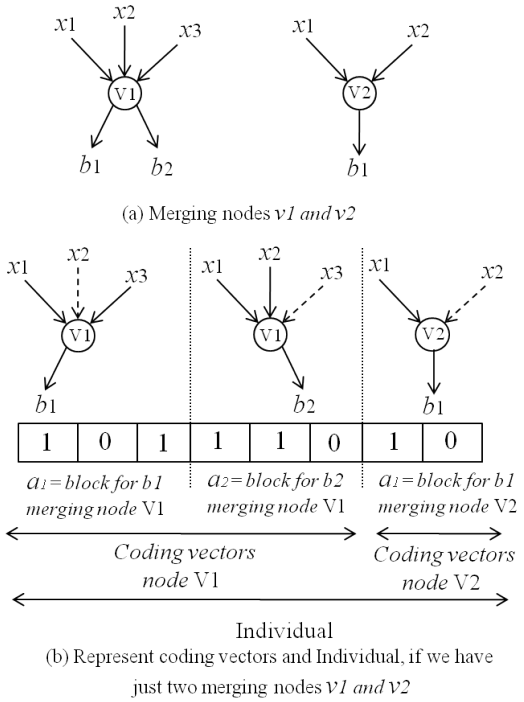

Fig.2. represent blocks of individuals for nodes *v1* and *v2* Final Stage

candidate solution $\left\{a_J\,\middle|J\in\tilde{E}\right\}$, we can construct a new graph on the basis of multigraph $G = (V, E)$ and candidate solution $\left\{a_J\,\middle|J\in\tilde{E}\right\}$ and then run max-flow from the source to each sink.

This new graph is constructed by decomposition of each merging node that is not a sink as follows:

Let us now decompose each merging node as follows to interpret network coding in the graph theoretic framework. Consider a node *v* with d_{in} (≥ 2) incoming links ($a_1,\ldots,a_{d_{in}}$) and d_{out} outgoing links ($b_1,\ldots,b_{dout}$). Let us first decompose the node by introducing d_{in} *incoming auxiliary nodes* ($k_1,\ldots,k_{d_{in}}$) and redirect node *v*'s d_{in} incoming links to each of the incoming auxiliary nodes (see Fig. 3). Similarly, we create d_{out} *outgoing auxiliary nodes* ($h_1,\ldots,h_{dout}$) and let node *v*'s d_{out} outgoing links to be the only outgoing links of each of the outgoing auxiliary nodes. We then introduce a link between each pair of incoming and outgoing auxiliary nodes [10].

Then, there is a one-to-one correspondence between the set of binary variables in coding vectors and the set of the introduced links between auxiliary nodes [10]. Fig. 3 shows this correspondence for node *v1* in Fig. 2.

After the decomposing procedure, we test the feasibility of the individual as following:, 1) deleting first, all the introduced links between auxiliary nodes associated with 0 in the Individual and 2) then calculating the max-flow between the source and each of the *d* sinks [10]. Note that by using decomposition method, all the outgoing auxiliary nodes with more than one input are considered as coding nodes. Having this, it's possible to get the maximum achievable multicast rate by finding the minimum of the individual max flow bounds between source and each of the links

If an outgoing auxiliary node has only one incoming link, we can delete that node and connect directly its incoming link

to its outgoing link without changing any of the max-flows. We also delete an auxiliary node which has no incoming link. If all d max-flows are at least R, rate R is achievable. We can calculate the fitness value (number of coding links) for each candidate by counting the number of remaining outgoing auxiliary nodes [10].

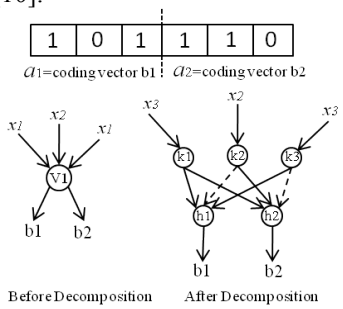

Fig.3. Decomposition of node $v1$ in Fig.2 with $d_{in} = 3$ and $d_{out} = 2$.

V. IMPROVEMENTS TO EDA

In this section, we discuss improvements we made to EDA for optimization problem (2). We discuss generating the initial population for problem (2) and then other two novel elements designed specifically for the problem.

A. *Generating the Initial Population*

The initial population is typically constructed randomly such that each component of the individual is assigned 0 or 1 with equal probabilities. Note, however, that the size of the population in EDA is usually much smaller than the size of the entire search space, and thus it is quite likely that randomly generated initial population may contain no feasible solution or very small number of feasible solutions. Our method of dealing with this problem is to include in the initial population many individuals that calls for many coding links. We make sure that some portions (e.g., x %) of the initial population consist of individuals $\{a_J | J \in \tilde{E}\}$ such that the number of vector components in $\{a_J | J \in \tilde{E}\}$ with value 1 exceeds some fraction (e.g., y %) of the total number of vector components. For sufficiently high percentages x and y, the initial population can include many feasible solutions, although their fitness values may not be good. Significant performance improvement was shown in EDA after employing this method.

B. *Block-Wise-Modified Representation and Operators*

Note that each candidate solution to optimization problem (2) is represented by a collection of binary vectors (strings) $\{a_J | J \in \tilde{E}\}$. We denote the dimension of vector $a_j, j \in \tilde{E}$ by k_j Then, the search space for optimization (2) is $\prod_{j \in \tilde{E}} 2^{k_j}$, including the infeasible candidates. However, we can reduce this search space. For an arbitrary link $J \in \tilde{E}$, as long as $\|a_J\|_1 \geq 2$, then link J contributes 1 to the cost function in (2) as a coding link. Among all strings with property $\|a_J\|_1 \geq 2$, the string of all 1's achieves the highest max-flow (1) in the graph constructed by decomposition described in section IV.B. Therefore, for the purpose of optimization (2), we only need to consider the string of all 1's in place of considering all strings with property $\|a_J\|_1 \geq 2$. Similarly, the string of all 0's and a string having only one element of value 1, both contribute none to the cost function of (2), but the latter achieves no less max-flow. Therefore, the string of all 0's need not to be considered. In our method, as candidate values of a_J, we only consider the string of all 1' and the strings having only single element of value 1. That is, we only consider strings, (100…0), (010…0), …., (00…01), and (11 1…1). This reduces the number of candidates as vector a_J to $k_J + 1$, and the size of the entire search space to $\prod_{j \in \tilde{E}} (k_j + 1)$. This methods significantly speed up our EDA algorithm, and this is also a slight improvement from "*Block-Wise* Representation and Operators" in [10], which results in the search space of size $\prod_{j \in \tilde{E}} (k_j + 2)$. We call our method "*Block-Wise-Modified* Representation and Operators".

C. *Greedy Sweep-Modified*

Greedy Sweep [10] is an operator that can be used to generate a number of more candidate solutions from the currently best candidate solution after evaluating all the members in the population in each iteration. It can be regarded as a local search. In [10], *Greedy Sweep* subjects each vector a_J, $j \in \tilde{E}$ in the currently best individual to its local search mechanism. We suggest modifying this method. In our modified method, which we call "*Greedy Sweep-modified*," we skip the vectors having only single element of value 1 in the Greedy Sweep operation.

VI. EXPERIMENTAL RESULTS

The parameters used for the experiments are as follows: Population size ($|\Delta_l|$) is 200 and the number of the best candidate solutions selected from set Δ_l is 10, which is 5% of population size. In the experiment results, we demonstrate the performance of EDA with *block-wise/Greedy Sweep* and *Block-Wise-Modified*/*Greedy Sweep-Modified* methods by carrying out simulations on various network topologies. We also compare our results with two previously proposed approaches by *M. Kim et al*. [9], [10] to minimize coding vectors. These methods use GA and are known as *Minimal 1* and *Minimal 2*. For *Minimal 1*, the algebraic method [9] is used without "*Block-Wise* representation and operators" and "*Greedy Sweep*" methods. For *Minimal 2*, the decomposition method is used with "*Block-wise* representation and operators" and "*Greedy Sweep*" methods. For each of the four methods, the best and the average values obtained from 20 random trials are shown in Table I. We use two types of network topologies for our simulations.

Network topology I: Suppose that link w in network A in Fig. 1 has capacity 2, which can be represented by two parallel links in the multi-graph. Fig. 4 illustrates a network in which a number of copies of network A are cascaded such that the source of each subsequent copy of A is replaced by an earlier copy of sink. This topology can be viewed as overlaying network A of Fig. 1 on each node of the binary tree in Fig. 4 b). We can expand the size of the network by overlaying network A on nodes of a larger binary treee.g., 31 copies of A and 4, 8, 16, and 32 sinks, respectively. We use

Table.1. Number of Coding Links Calculated

		Network topology I								Network topology II			
		3 copies		7copies		15 copies		31 copies		(50,87,10,5)		(75, 156,15,7)	
		Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
EDA	Block-wise-mod /with Greedy Sweep-mod (Num. of Iter.)	0.0/ 0.0 (21)	0.0/ 0.0 (18)	0.0/ 0.0 (25)	0.0/ 0.0 (20)	0.0/ 0.0 (48)	1.22/ 1.03 (55)	3/ 2 (98)	4.09/ 3.16 (90)	1/ 1 (57)	1.79/ 1.67 (71)	2/ 2 (102)	2.46/ 2.25 (94)
	Block-wise /with Greedy Sweep (Num. of Iter.)	0.0/ 0.0 (18)	0.0/ 0.0 (23)	0.0/ 0.0 (22)	0.0/ 0.0 (22)	1/ 0.0 (51)	1.91/ 1.88 (60)	3/ 2 (80)	4.66/ 3.94 (99)	2/ 2 (66)	2.20/ 2.10 (75)	2/ 2 (132)	3.23/ 3.18 (122)
GA	Minimal 1 (Num. of Iter.)	0.0 (100)	0.65 (100)	0.0 (100)	1.15 (100)	2 (200)	3.35 (200)	6 (500)	8.2 (500)	5 (500)	6.42 (500)	7 (500)	8.22 (500)
	Minimal 2 (Num. of Iter.)	0.0 (100)	0.0 (100)	0.0 (100)	0.0 (100)	1 (200)	2.12 (200)	4 (500)	6.87 (500)	2 (500)	2.40 (500)	3 (500)	3.63 (500)

binary trees containing 3, 7 (as illustrated by Fig. 4 c), 15, andthese networks for our simulation. In all these networks, the maximum multicast rate 2 is achievable without coding, so the minimal number of coding links that achieve multicast rate 2 is zero. Therefore, the minimum value of 0 for optimization (2) can be used as a benchmark.

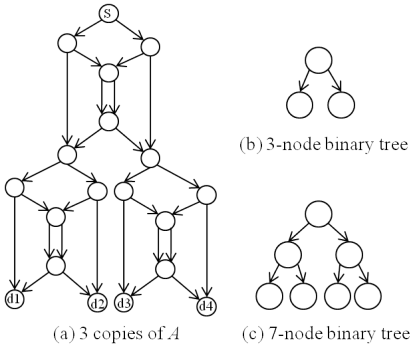

Fig. 4. Network toplogy I

Network topology II: In the second set of sample networks, two topologies generated by the algorithm in [14] are employed with the following parameters: (50 nodes, 87 links, 10 sinks, rate 5) and (75 nodes, 156 links, 15 sinks, rate 7) employed.

In our experiments, we have presented the performance of EDA and GA and their improvements (such as *Greedy Sweep Modified*, *Block-wise-modified* Representation, etc.) Table I. shows that the *block-wise-modified* representation and operators, clearly outperform the *block-wise* counterpart in all aspects. It is observed that the EDA with *block-wise-modified* and *Greedy Sweep-Modified* outperform the EDA with *Block-wise* and *Greedy Sweep*. We can also observe that the performance of EDA is at least as good and often better than that of both Minimal 1 [9] and Minimal 2 [10] both in terms of the best and the average values (the number of coding links). More impressively, EDAs produce solutions with a smaller number of coding links and a smaller number of iterations than GAs (Minimal 1 and Minimal 2).

Our proposed EDA method outperforms the algorithms introduced in Minimal 1 and Minimal 2 methods, while providing a comparable computational complexity.

VII. Conclusions and Future Work

In this paper, we presented EDAs for minimizing network coding resources problem. The simple model, low implementation complexity, and resistance to being trapped in a local minimum, all make EDA a suitable candidate for solving the problem of minimizing network coding resources.

There are several topics for further research. i.e., we may further improve the centralized algorithm, by a smarter management of population such that it converges faster to a better solution. EDA components of the proposed approach, such as the method for constructing the initial population, can be further specialized for the problem at hand to improve the algorithm's performance.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

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

[3] D. Petrovic, K. Ramchandran, and J. Rabaey, "Overcoming Untuned Radios in Wireless Networks with Network Coding", in *IEEE Transactions on Information Theory*, Vol 52, No. 6, pp. 2649-2657, 2006.

[4] C. Gkantsidis and P. Rodriguez, "Network Coding for Large Scale File Distribution", in Proc. IEEE INFOCOM, 2005.

[5] S. Li, R. Yeung, and N. Cai, "Linear Network Coding", in IEEE Transactions on Information Theory, Vol 49, No. 2, pp. 371381, 2003.

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

[7] M. Langberg, A. Sprintson, and J. Bruck, "The encoding complexity of network coding," IEEE Trans. Inf. Theory, vol. 52, no. 6, pp. 2386–2397, 2006.

[8] C. Fragouli and E. Soljanin, "Information flow decomposition for network coding," IEEE Trans. Inform. Theory, vol. 52, pp. 829-848, March 2006.

[9] M. Kim, C. W. Ahn, M. M´edard, and M. Effros, "On minimizing network coding resources: An evolutionary approach," in Proc. NetCod, 2006.

[10] M. Kim, M. M´edard, V. Aggarwal, U. O'Reilly, W. Kim, C. Wook Ahn, and M. Effros "Evolutionary Approaches To Minimizing Network Coding Resources" IEEE INFOCOM 2007.

[11] A.E. Eiben and J.E. Smith, *Introduction to Evolutionary Computing,* Springer Verlag, 2003.

[12] P. Larrañaga and J. A Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.

[13] M. Naeem and D.C. Lee, " Estimation of Distribution Algorithm for Scheduling in Uplink Multiuser Wireless Communication System," *IEEE Symposium on Computational Intelligence in Scheduling* , March 2009, pp 36 – 41, TN, USA

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

[15] Michalewicz, Z., Schoenauer, M.: Evolutioary computation for constrained parameter optimization problems. Evolutionary Computation 4 (1996) 1-32.