A Population Based Incremental Learning for Network Coding Resources Minimization

Huanlai Xing and Rong Qu

***Abstract*—In network coding based multicast, coding operations need to be minimized as they consume computational resources and increase data processing complexity at corresponding nodes in the network. To address the problem, we develop a population based incremental learning algorithm which shows to outperform existing algorithms in terms of both the solution obtained and computational time consumed on networks with various features.**

***Index Terms*—multicast, network coding, population based incremental learning.**

I. Introduction

In network coding, each intermediate node in the network is allowed, if necessary, to recombine data packets received from different incoming links [1]. By doing so, a maximized multicast throughput can always be achieved [2]. Due to the necessary coding operations, network coding incurs additional cost such as computational overhead or transmission delay [3-4]. It is thus important that the amount of coding operations is minimized while the benefits of network coding are warranted. However, such problem is NP-hard [4].

To tackle the problem, several evolutionary algorithms are proposed in the literature. In [4], a genetic algorithm (GA) in an algebraic framework is applied to acyclic networks. The GA is then extended to networks with cycles by a graph decomposition method [5]. In [6], two GAs with different genotype encodings and operators are compared. A quantum-inspired evolutionary algorithm (QEA) is developed in our previous work, showing to outperform simple GA [7]. However, as we observed in this paper, the results obtained by QEA are sometimes suboptimal, and this motivates us to develop a more effective algorithm.

Population based incremental learning (PBIL) is a combination of GA and competitive learning and has many applications [8-10]. Different from GA, PBIL maintains a real-valued probability vector, rather than a population, thus yields a much lower memory requirement. Besides, with no complex genetic operator such as crossover, PBIL occurs much less computational cost. In this letter, we present an effective PBIL to minimize the amount of coding operations required.

II. Problem Formulation

A communication network can be modeled as a directed graph $G = (V, E)$, where V and E denote the sets of nodes and links, respectively [2]. A single-source network coding based multicast can be defined as a set (G, s, T, R), where information needs to be transmitted at data rate R from the source $s \in V$ to a set of sinks $T = \{t_1, ..., t_d\} \subset V$ in the graph G. Data rate R is achievable if there is a routing scheme that enables each t_k, $k = 1, ..., d$, to receive information at rate R [4-6]. We assume that each link has a unit capacity, and a path from s to t_k thus has a unit capacity. If we manage to set up R link-disjoint paths $\{p_1(s, t_k), ..., p_R(s, t_k)\}$ from s to each $t_k \in T$, we make rate R achievable.

A subgraph is referred to as *network coding based multicast subgraph* (NCM subgraph, denoted by G_{NCM}) if there are R link-disjoint paths $p_i(s, t_k)$, $i = 1, ..., R$, from s to each sink t_k, $k = 1, ..., d$, within this subgraph. To find a G_{NCM}, we first find R link-disjoint paths for each $t_k \in T$ from G, i.e. $p_1(s, t_k), ..., p_R(s, t_k)$. We then select a subgraph G_{NCM} from G that only contains these $R \cdot d$ paths. No coding happens at a node with one incoming link. We refer to a non-sink node with multiple incoming links as a merging node [5-6]. We refer to each outgoing link of a merging node as a potential coding link. A potential coding link becomes coding link if this link is dependent on at least two of its incoming links.

For a given multicast, the number of coding links is more precise to indicate the total amount of coding operations [3]. We therefore investigate to find a NCM subgraph G_{NCM} with coding links minimized. We define the following notations:

σ_{ij}: a variable associated with the j-th outgoing link of the i-th merging node, $i = 1, ..., M$, $j = 1, ..., Z_i$, where M is the total number of merging nodes and the i-th merging node has Z_i outgoing links. $v_{ij} = 1$ if the j-th outgoing link of the i-th node serves as a coding link; $v_{ij} = 0$ otherwise.

$\Phi(G_{NCM})$: the number of coding links in a given NCM subgraph G_{NCM}.

$\lambda(s, t_k)$: the achievable rate from s to t_k in G_{NCM}.

We define the problem concerned as to find a NCM subgraph G_{NCM} where the number of coding links, $\Phi(G_{NCM})$, is minimized and the data rate R is met, shown as follows:

Mimimize:

$$\Phi(G_{NCM}) = \sum_{i=1}^{M} \sum_{j=1}^{Z_i} \sigma_{ij} \tag{1}$$

subject to:

$$\lambda(s, t_k) = R, \ \forall t_k \in T \tag{2}$$

Objective (1) defines our problem as to minimize the number of coding links; Constraint (2) defines that the achievable data rate between s and each sink is R.

Manuscript received February 21, 2011. The associate editor coordinating the review of this letter and approving it for publication was A. Burr.

This work was supported in part by China Scholarship Council, China, and The University of Nottingham, UK.

The authors are with Automated Scheduling, Optimisation and Planning (ASAP) Group, The School of Computer Science, The University of Nottingham, Nottingham NG8 1BB, UK (e-mail: {hxx, rxq}@cs.nott.ac.uk).

Digital Object Identifier 10.1109/LCOMM.2011.11.110274

III. The Population based Incremental Learning

PBIL maintains a real-valued probability vector which, when sampled, generates promising solutions with higher probability [8]. Let $\mathbf{P}^{(t)} = \{P_1, P_2, ..., P_L\}$ be a probability vector at generation t that generates binary solutions, where P_i is the probability of obtaining '1' at the i-th position and L is the solution length. At each generation, $\mathbf{P}^{(t)}$ is sampled N times to obtain N samples (i.e. solutions). Among them, high-quality samples are selected and their statistic information is used to adjust $\mathbf{P}^{(t)}$. Initially, the value of P_i is set to 0.5, $i = 1$, ..., L, and the first N samples are randomly created from the solution space. As search progresses, $\mathbf{P}^{(t)}$ is gradually shifted towards an explicit solution as P_i, $i = 1$, ..., L, is approaching to either 0.0 or 1.0.

A. Probability Vector Update Scheme

Our PBIL uses a Hebbian-inspired rule [11] to update $\mathbf{P}^{(t)}$ as described below. At generation t, the best ever found solution $\mathbf{B}^{(t-1)}$ is inserted into the N obtained samples. The $N+1$ solutions are then ordered by their fitness and the μ ($\mu \leq N+1$) best solutions, $\mathbf{Y}_{1:N+1}$, ..., $\mathbf{Y}_{\mu:N+1}$, are selected. The statistic information of the μ solutions, i.e. $\mathbf{P}_{select}$, is extracted and used to modify $\mathbf{P}^{(t)}$. Note that $\mathbf{P}^{(t)}$ produces feasible solutions as well as infeasible ones [5-6]. Let $\rho^{(t)}$ be the number of feasible solutions among the $N+1$ solutions at generation t, $\mu^{(t)}$ be the number of selected solutions at generation t, and LR be the learning rate. The update scheme at generation t is as follows:

$$\mathbf{P}^{(t)} = (1.0 - LR) \cdot \mathbf{P}^{(t-1)} + LR \cdot \mathbf{P}_{select} \tag{3}$$

$$\mathbf{P}_{select} = \frac{1}{\mu^{(t)}} \cdot \sum_{k=1}^{\mu^{(t)}} \mathbf{Y}_{k:N+1} \tag{4}$$

$$\mu^{(t)} = \begin{cases} \alpha \cdot N, & if\ \rho^{(t)} \geq \alpha \cdot N \\ \rho^{(t)}, & otherwise \end{cases} \tag{5}$$

where α indicates the maximum proportion of solutions to be selected. In this paper, $\alpha = 20\%$.

In this scheme, the statistics of selected solutions are used to update $\mathbf{P}^{(t)}$ and limits the contribution to $\mathbf{P}^{(t)}$ from any single solution. This helps to prevent $\mathbf{P}^{(t)}$ from converging rapidly to local optima and thus to maintain an appropriate diversification.

B. Restart Scheme

During the search, $\mathbf{P}^{(t)}$ gradually converges to an explicit solution. If $\mathbf{P}^{(t)}$ is shifted to a local optima and the uncertainty of $\mathbf{P}^{(t)}$ is low enough, the search will be stuck to this local optima. We therefore restart the search in PBIL once the uncertainty of $\mathbf{P}^{(t)}$ is lower than a threshold. Inspired by the work in [9], the average entropy per bit of $\mathbf{P}^{(t)}$, E_P, is introduced to measure the uncertainty, as shown below:

$$E_P = -\frac{1}{L} \sum_{i=1}^{L} \sum_{k=0}^{1} P_i^{(x=k)} \cdot ln(P_i^{(x=k)}) \tag{6}$$

where $P_i^{(x=k)}$ denotes the probability of generating 'x' at the i-th position, $P_i^{(x=0)} = 1 - P_i$ and $P_i^{(x=1)} = P_i$.

It can be seen that E_P decreases and approaches to 0 when $\mathbf{P}^{(t)}$ is converging to an explicit solution during the evolution. Through empirical experiments we observed that $\mathbf{P}^{(t)}$ can hardly find a better solution when P_i is either higher than 0.97 or lower than 0.03. The critical value of E_P is thus calculated as 0.134, i.e. when each P_i is either 0.97 or 0.03. Let Ω denote the critical value of E_P. In experiments, we set $\Omega = 0.14$. If $E_P < \Omega$, PBIL restarts from the same initial solutions as those of the very beginning of the algorithm.

C. The Procedure of the PBIL Algorithm

1) Set generation $t = 0$
2) **Initialization**
3) For $i = 1$, 2, ..., L, set $P_i = 0.5$
4) Set $\mathbf{B}^{(t)}$ as an all-one solution and evaluate $\mathbf{B}^{(t)}$
5) Generate N samples from $\mathbf{P}^{(t)}$
6) **Repeat**
7) Set $t = t + 1$
8) Evaluate the N samples
9) Select $\mu^{(t)}$ solutions from the N samples and $\mathbf{B}^{(t)}$
10) Calculate $\mathbf{P}_{select}$ by using Eq.(4)
11) Update $\mathbf{P}^{(t)}$ by Eq.(3)
12) **If** $E_P < \Omega$ **then**
13) For $i = 1$, 2, ..., L, reset $P_i = 0.5$
14) Reset $\mathbf{B}^{(t)}$ as an all-one solution and evaluate $\mathbf{B}^{(t)}$
15) Generate N samples from $\mathbf{P}^{(t)}$
16) **Until** *termination condition is met*

In the PBIL framework, we represent the problem being concerned by using the graph decomposition method proposed in [5]. Binary link state (BLS) [6] is adopted as the encoding scheme. In the fitness evaluation, each infeasible solution is assigned a sufficiently large value (50 in this paper). If a solution is feasible, i.e. it corresponds to a valid NCM subgraph, its fitness value is given by the number of coding links in the NCM subgraph. To ensure that the algorithm begins with at least one feasible solution in the population, the initial best solution is set as an all '1' vector where all merging nodes are active [5-6]. The termination criteria are subject to two conditions: 1) a NCM subgraph without coding is obtained, or 2) the algorithm reaches a pre-defined number of generations.

IV. Numerical Results and Discussions

We compare the following five algorithms:

1) GA-1: GA with block transmission state encoding and operators [6].

2) GA-2: GA with binary link state encoding and operators [6].

3) QEA: quantum inspired evolutionary algorithm [7].

4) PBIL-1: the proposed PBIL without restart scheme.

5) PBIL-2: the proposed PBIL with restart scheme.

Simulations have been carried out upon two fixed and six random networks, where optimal solutions exist that yield NCM subgraphs without coding. The two fixed networks are 7-copies network (57nodes, 84links, 8sinks, $R = 2$) and 15-copies network (121nodes, 180links, 16sinks, $R = 2$). The six random networks include Rand-1 (40nodes, 78links, 9sinks,

TABLE I
NUMERICAL RESULT COMPARISONS

	7-copies [6]				15-copies [6]				Rand-1				Rand-2			
Algorithm	SR (%)	AVG	SD	ACT (sec.)	SR (%)	AVG	SD	ACT (sec.)	SR (%)	AVG	SD	ACT (sec.)	SR (%)	AVG	SD	ACT (sec.)
GA-1 [6]	95	0.05	0.2	35.7	5	2.2	1.5	213.2	30	0.9	0.7	48.6	100	0.0	0.0	5.3
GA-2 [6]	90	0.1	0.3	24.1	10	2.6	1.4	209.4	30	0.8	0.6	46.1	100	0.0	0.0	6.6
QEA [7]	50	1.2	2.4	45.5	0	7.3	6.6	272.2	55	0.5	0.6	32.0	100	0.0	0.0	1.1
PBIL-1	100	0.0	0.0	4.7	75	0.3	0.5	245.6	100	0.0	0.0	8.5	100	0.0	0.0	0.2
PBIL-2	100	0.0	0.0	3.5	100	0.0	0.0	70.8	100	0.0	0.0	5.6	100	0.0	0.0	0.4
	Rand-3				**Rand-4**				**Rand-5**				**Rand-6**			
Algorithm	SR (%)	AVG	SD	ACT (sec.)	SR (%)	AVG	SD	ACT (sec.)	SR (%)	AVG	SD	ACT (sec.)	SR (%)	AVG	SD	ACT (sec.)
GA-1 [6]	25	0.8	0.5	72.6	10	1.7	1.0	108.9	10	1.4	0.8	176.9	10	1.2	0.6	188.2
GA-2 [6]	15	0.9	0.5	75.3	5	1.9	0.9	113.0	0	2.1	1.0	184.7	10	1.4	0.7	224.5
QEA [7]	50	0.5	0.5	55.1	50	0.6	0.6	161.2	10	1.6	1.1	204.2	70	0.3	0.4	113.8
PBIL-1	95	0.05	0.2	25.5	95	0.05	0.2	49.3	95	0.05	0.2	97.5	100	0.0	0.0	56.3
PBIL-2	100	0.0	0.0	25.6	100	0.0	0.0	28.3	100	0.0	0.0	62.2	100	0.0	0.0	38.5

$R = 3$), Rand-2 (40nodes, 85links, 9sinks, $R = 4$), Rand-3 (50nodes, 101links, 8sinks, $R = 3$), Rand-4 (50nodes, 118links, 10sinks, $R = 4$), Rand-5 (60nodes, 150links, 11sinks, $R = 5$) and Rand-6 (60nodes, 156links, 10sinks, $R = 4$). In all algorithms, the population size and pre-defined termination generation are set to 40 and 500, respectively. The crossover probability, tournament size and mutation probability are set to 0.8, 10 and 0.012 for GA-1, and 0.8, 10 and 0.006 for GA-2, respectively. Please refer to [7] for parameters settings of QEA. In PBIL-1 and PBIL-2, $LR = 0.1$ and $\alpha = 0.2$. In PBIL-2, $\Omega = 0.14$. By running each algorithm 20 times, we collected the successful ratio of finding a subgraph without coding (SR), the average best fitness (AVG) and standard deviation (SD), and the average computational time (ACT), as shown in Table I. All simulations were run on a Windows XP computer with Intel(R) Core(TM)2 Duo CPU E8400 3.0GHz, 2G RAM.

It can be seen clearly that PBIL-2 outperforms the other four algorithms. In terms of successful ratio, PBIL-2 obtains 100% in each instance, always being able to find optimal solutions in each run. PBIL-1 is the second best, with a higher SR than GA-1, GA-2 and QEA in every instance. The differences between the successful ratio of these existing algorithms and the two PBILs reveal that our PBILs have better global exploration ability. Besides, the results of average best fitness and standard deviation also demonstrate that PBILs perform better than GA-1, GA-2 and QEA. In addition, it is evident that PBILs, especially PBIL-2, consumes less average computational time in all instances. These all indicate that the proposed PBIL is more effective to solve coding resource optimization problem. Besides, QEA shows better optimization performance than GA-1 and GA-2 in most of the instances. However, the performance of QEA shows to be sensitive to instances and sometimes cannot find optimal solutions. When comparing the performance of PBIL-1 and PBIL-2, the latter achieves better optimization results with less computing time. This is because the restart scheme also helps to improve the global searching ability of PBIL-2. PBIL-2 can promptly approach to one of optimal solutions and it stops once an optimal solution is found regardless of the pre-defined generation not being reached (see the stopping condition).

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, 2000.

[2] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[3] 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.

[4] M. Kim, C. W. Ahn, M. Medard, *et al*., "On minimizing network coding resources: an evolutionary approach," in *Proc. NetCod 2006*.

[5] M. Kim, M. Medard, V. Aggarwal, *et al*., "Evolutionary approaches to minimizing network coding resources," in *Proc. INFOCOM 2007*, pp. 1991–1999.

[6] M. Kim, V. Aggarwal, and V. O. Reilly, *et al*., "Genetic representations for evolutionary optimization of network coding," in *Proc. EvoWorkshops 2007*, pp. 21–31.

[7] H. Xing, Y. Ji, L. Bai, *et al*., "An improved quantum-inspired evolutionary algorithm for coding resource optimization based network coding multicast scheme," *Int. J. Electron. Commun.*, vol. 64, no. 12, pp. 1105–1113, 2010.

[8] S. Baluja, "Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning," Technical Report CMU-CS-94-163, Carnegie Mellon University, 1994.

[9] H. Pang, K. Hu, and Z. Hong, "Adaptive PBIL algorithm and its application to solve scheduling problems," in *Proc. ISIC 2006*, pp. 784–789.

[10] H. Xing and R. Qu, "A population based incremental learning for delay constrained network coding resource minimization," in *Proc. EvoApplications 2011*, pp. 51–60.

[11] C. Gonzalez, J. A. Lozano, and P. Larranaga, "Analyzing the population based incremental learning algorithm by means of discrete dynamical systems," *Complex Systems*, vol. 12, no. 4, pp. 465-479, 2000.