A Population Based Incremental Learning for Delay Constrained Network Coding Resource Minimization

Huanlai Xing and Rong Qu

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

Abstract. In network coding based multicast, coding operations are expected to be minimized as they not only incur additional computational cost at corresponding nodes in network but also increase data transmission delay. On the other hand, delay constraint must be concerned particularly in delay sensitive applications, e.g. video conferencing. In this paper, we study the problem of minimizing the amount of coding operations required while meeting the end-to-end delay constraint in network coding based multicast. A population based incremental learning (PBIL) algorithm is developed, where a group of best so far individuals, rather than a single one, is maintained and used to update the probability vector, which enhances the global search capability of the algorithm. Simulation results demonstrate the effectiveness of our PBIL.

Keywords: multicast; network coding; population based incremental learning

1 Introduction

Network coding has been attracting an increasing research attention since 2000 [1]. Instead of simply replicating and forwarding data packets at the network layer, network coding allows any intermediate node, if necessary, to perform mathematical operations to recombine data packets received from different incoming links. By doing so, the maximized multicast throughput is always obtained [2].

In most of the previous research on network coding, coding is performed at all coding-possible nodes without concerning issues raised in real life applications. One such issue is that, to obtain an expected multicast throughput, coding may only be necessary at a subset of coding-possible nodes [3,4,5]. As they consume public resources, e.g. buffer and computational resources, coding operations should be minimized to leave more public resources for other network applications. Another issue in real time applications is that transmission delays, especially in delay sensitive applications, e.g. video conferencing and distributed game, should be bounded. It is therefore important to minimize coding operations while meeting the end-to-end delay requirement. However, such problem has not drawn enough research attention.

A number of algorithms have been proposed to minimize coding resources, however, without concerning the delay constraint. In [6] and [7], original topologies were decomposed and transformed into secondary graphs, based on which greedy

algorithms were developed to reduce the amount of coding operations. In [8], linear programming formulations based on a model allowing continuous flows were proposed to optimize various resources used for network coding. In addition, Kim *et al* investigated centralized and distributed genetic algorithms (GAs) with problem-specific operators to minimize the required network coding resources [3,4,5].

Population based incremental learning (PBIL), a combination of evolutionary algorithm and competitive learning, was first introduced in 1994 [9]. Without using crossover and mutation, PBIL retains the stochastic search nature of GA by simply maintaining a probability vector. Since its introduction, PBIL has shown to be very successful on numerous benchmark and real-world problems [10].

In this paper, we propose the first PBIL algorithm for the problem of minimizing network coding resources with the delay bound in network coding based multicast. We put forward a new probability vector update scheme based on statistical information of a group of, rather than a single, best so far individuals. In addition, we observed that the use of the all-one vector in the initialization of PBIL greatly helps to guide the search towards promising solutions. Simulation results demonstrated that our PBIL is highly effective for the problem concerned.

2 Problem Description

A communication network can be modeled as a directed graph $G = (V, E)$, where V and E denote the set of nodes and links, respectively [2]. We assume each link e has a unit capacity. A delay constrained single-source network coding based multicast scenario can be defined as a 5-tuple set (G, s, T, R, Ω), where the information needs to be transmitted at the data rate R from the source node $s \in V$ to a set of sinks $T = \{t_1,\ldots,t_d\} \subset V$ in the graph $G(V, E)$, satisfying a given end-to-end delay constraint Ω. Rate R is achievable if there is a transmission scheme that enables each sink t_k, $k = 1,\ldots,d$, to receive information at data rate R [4,5]. As each link has a unit capacity, a path from s to t_k thus has a unit capacity. If we manage to set up R link-disjoint paths $\{P_1(s, t_k),\ldots,P_R(s, t_k)\}$ from s to each sink $t_k \in T$, we make the data rate R achievable. We concern linear network coding which is sufficient for multicast [2].

In this paper, a subgraph is referred to as a *network coding based multicast subgraph* (NCM subgraph, denoted by $G_{NCM}(s, T)$) if there are R link-disjoint paths $P_i(s, t_k)$, $i = 1,\ldots,R$, from s to each sink t_k in this subgraph. We refer to a non-sink node with multiple incoming links as a *merging node* [4,5]. Only merging nodes and their outgoing coding links have the potential to serve as coding nodes and coding links, respectively [4,5]. To determine if an outgoing link of a merging node becomes a coding link, we only need to check if the information via this link is dependent on at least two incoming links of the merging node.

For a given multicast scenario, the number of coding links is more precise to indicate the total amount of coding operations [7]. We hereinafter investigate how to construct a NCM subgraph $G_{NCM}(s, T)$ with the minimal number of coding links while achieving the expected data rate and satisfying the end-to-end delay constraint Ω.

We define the following notations in our paper:

- x_{ij}: a variable associated with the j-th outgoing link of the i-th merging node, $i = 1,\ldots,M$, $j = 1,\ldots,Z_i$, where M is the total number of merging nodes and the i-th node has Z_i outgoing links. $x_{ij} = 1$ if the j-th outgoing link of the i-th node serves as a coding link; $x_{ij} = 0$ otherwise.
- $N_{cl}(G_{NCM}(s,T))$: the number of coding links in a NCM subgraph $G_{NCM}(s, T)$.
- $R(s, t_k)$: the achievable rate from s to t_k.
- R: the defined data rate (an integer) at which s expects to transmit information.
- $e(n_a, n_b)$: a directed link from node n_a to node n_b.
- $P_i(s, t_k)$: the i-th established path from s to t_k, $i = 1,\ldots,R$ in $G_{NCM}(s, T)$.
- $W_i(s, t_k) = \{e \mid e \in P_i(s, t_k)\}$: the link set of $P_i(s, t_k)$.
- $P_i^{s \to t_k}(n_a, n_b)$: the subpath from node n_a to node n_b on the path $P_i(s, t_k)$. Along a path, we assume that node n_{j-1} is the parent node of node n_j, $j = 2,\ldots,p$, $n_1 = s$ and $n_p = t_k$.
- $D(\omega)$: the delay of the term ω. $D(\omega)$ represents the data processing delay of a node ω; or it is the propagation delay of a link ω. If ω is a path $P(n_i,n_j)$, $D(\omega)$ is the end-to-end delay from node n_i to node n_j; if ω is a NCM subgraph $G_{\mathrm{NCM}}(s,T)$, $D(\omega)$ denotes the maximal path delay in $G_{\mathrm{NCM}}(s,T)$.

Based on the above notations, we define the problem of delay constrained coding resource minimization as to minimize the number of coding links while achieving the desired multicast throughput and meeting the delay restriction, shown as follows:

$$\textit{Minimize}: \qquad N_{cl}(G_{NCM}(s,T)) = \sum_{i=1}^{\mathrm{M}} \sum_{j=1}^{\mathrm{Z}_i} x_{ij} \tag{1}$$

$$\textit{Subject to}: \qquad R(s,\ t_k) \geq R,\ \ \forall t_k \in T \tag{2}$$

$$\bigcap_{i=1}^{R} W_i(s,t_k) = \varnothing,\ \ \forall t_k \in T \tag{3}$$

$$D(G_{NCM}(s,\ T)) \leq \Omega \tag{4}$$

where,

$$D(G_{NCM}(s,\ T)) = \max\{D(P_i(s,\ t_k)) \mid i = 1,\ 2,\ \ldots,\ R,\ \forall t_k \in T\} \tag{5}$$

Objective (1) defines our problem as to find a NCM subgraph with the minimal number of coding links; Constraint (2) defines that the practical achievable data rate from s to every sink must be at least R so that R paths can be constructed for each sink; Constraint (3) restricts that for each t_k the R constructed paths $P_i(s, t_k)$, must have no common link; Constraint (4) defines that the maximal delay of the constructed NCM subgraph cannot be greater than the pre-defined delay bound Ω; The delay of the NCM subgraph is defined in (5) as the maximal delay among all paths. Note that in this paper we only consider the transmission delay from s to t_k along each path $P_i(s, t_k)$. The decoding delay to obtain the original information in each sink is omitted.

Along a path $P_i(s, t_k)$, there are in general four types of nodes: the source, the sink, forwarding node(s) and coding node(s). If a node n_j is a sink or a forwarding node, we

ignore its data processing delay, i.e. $D(n_j) = 0$. The delay of the path from s to the sink or the forwarding node is thus defined as follows:

$$D(P_i^{s\to t_k}(s, n_j)) = D(P_i^{s\to t_k}(s, n_{j-1})) + D(e(n_{j-1}, n_j)) \tag{6}$$

If a number of data packets are required to be coded together at a coding node, coding operation cannot start until the arrival of the last data packet. If a node n_j is a coding node, we assume that among the $R{\cdot}d$ paths in $G_{NCM}(s, T)$ there are Y paths that pass through n_j. Obviously, along each of the Y paths there is a subpath from s to n_j. We denote the delays of the Y subpaths by $D_1(s, n_j)$, $D_2(s, n_j)$, …, $D_Y(s, n_j)$. The delay of the subpath from s to coding node n_j is defined as follows:

$$D(P_i^{s\to t_k}(s, n_j)) = \max\{D_r(s, n_j) \mid r = 1, ..., Y\} + \Delta_c \tag{7}$$

where Δ_c is the time consumed by the coding operation. We assume any coding operation consumes the same time Δ_c.

3 The Proposed PBIL

As an estimation of distribution algorithm, PBIL maintains a real-valued probability vector which, when sampled, generates promising solutions with high probabilities. The procedure of the standard PBIL with elitism [9] is shown in Fig.1. In the probability vector $\boldsymbol{P}(t) = \{P_1^t, P_2^t, \ldots, P_L^t\}$ at generation t, where L is the solution length, and P_i^t, $i = 1, 2, \ldots, L$, is the probability of generating '1' at the i-th position of a solution and is initialized as 0.5. At each generation, $\boldsymbol{P}(t)$ is sampled to form a set $S(t)$ of N individuals (i.e. solutions) which are evaluated and assigned a fitness value using a given fitness function. The best so far individual $\boldsymbol{B}(t) = \{B_1^t, B_2^t, \ldots, B_L^t\}$ is then selected to update $\boldsymbol{P}(t)$ as follows:

$$P_i^t = (1.0 - \alpha) \cdot P_i^t + \alpha \cdot B_i^t, \quad i = 1, 2, \ldots, L \tag{8}$$

where α is the learning rate.

After the $\boldsymbol{P}(t)$ is updated, a bit-wise mutation operation may be adopted to maintain the diversity and avoid local optima [9]. We denote by pm and σ the mutation probability and the amount of mutation at each locus P_i^t, respectively. For each locus P_i^t, a uniformly distributed random number rnd_i in [0.0, 1.0] is generated. If $rnd_i < pm$, P_i^t is mutated by the following formula:

$$P_i^t = (1.0 - \sigma) \cdot P_i^t + rnd(\{0.0, 1.0\}) \cdot \sigma \tag{9}$$

where $rnd(\{0.0, 1.0\})$ is 0.0 or 1.0, randomly generated with a probability 0.5.

After mutation, a sampling set is generated by the new $\boldsymbol{P}(t)$. Step 6 to 11 is repeated until the termination condition is met. Along with the evolution, $\boldsymbol{P}(t)$ gradually converges to an explicit solution.

In this paper, we propose a new probability vector update scheme where a number of best so far individuals are adopted. In addition, an all-one vector is employed at the beginning of the algorithm to guide the search towards feasible solutions.

Standard PBIL with elitism
1) **Initialization**
2) Set $t = 0$;
3) ***for*** $i = 1$ to L, ***do*** set $P_i^t = 0.5$
4) Generate a sampling set $S(t)$ of N individuals from $\boldsymbol{P}(t)$
5) **repeat**
6) Set $t = t + 1$
7) Evaluate the samples in $S(t\text{-}1)$
8) Find the best individual $\boldsymbol{B}(t)$ from $\boldsymbol{B}(t\text{-}1) \cup S(t\text{-}1)$
9) Update $\boldsymbol{P}(t)$ by Eq.(8)
10) Mutate $\boldsymbol{P}(t)$ by Eq.(9)
11) Generate set $S(t)$ by sampling $\boldsymbol{P}(t)$
12) **until** *termination condition is met*

Fig. 1. Procedure of PBIL

The new probability vector update scheme
1) Find the H best individuals from $S(t\text{-}1)$ and sort them in sequence $\{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_H\}$, where $f(\boldsymbol{C}_1) \le f(\boldsymbol{C}_2) \le \ldots \le f(\boldsymbol{C}_H)$
2) ***for*** $i = 1$ to H ***do***
3) Find the worst individual $\boldsymbol{B}_{MAX}$ in the set of best so far individuals S_{BSF}, where $f(\boldsymbol{B}_{MAX}) = \max\{f(\boldsymbol{B}_1), f(\boldsymbol{B}_2), \ldots, f(\boldsymbol{B}_H)\}$
4) ***if*** $f(\boldsymbol{C}_i) \le f(\boldsymbol{B}_{MAX})$ ***do***
5) $\boldsymbol{B}_{MAX} = \boldsymbol{C}_i$;
6) $f(\boldsymbol{B}_{MAX}) = f(\boldsymbol{C}_i)$;
7) end ***if***
8) end ***for***
9) Update $\boldsymbol{P}(t)$ by Eq.(10) and Eq.(11)

Fig. 2. The new probability update scheme

3.1 The New Probability Vector Update Scheme

Contrary to the traditional PBIL [9] that updates $\boldsymbol{P}(t)$ by shifting it towards the best individual $\boldsymbol{B}(t)$, our PBIL concerns a set of best so far individuals $S_{BSF} = \{\boldsymbol{B}_1, \ldots, \boldsymbol{B}_H\}$, where $H \ge 1$ is a constant number. Initially, $\boldsymbol{P}(t)$ is sampled H times to create H individuals to form S_{BSF}. At each generation, when a number of fitter individuals appear, we update S_{BSF} by replacing those with worse fitness values in S_{BSF} by the fitter ones. Then, the statistical information of S_{BSF}, i.e. $\boldsymbol{P}_{BSF}$, is extracted and used to update $\boldsymbol{P}(t)$, as shown in formula (10) and (11).

$$\boldsymbol{P}_{BSF} = \frac{1}{H} \cdot \sum_{k=1}^{H} \boldsymbol{B}_k \tag{10}$$

$$\boldsymbol{P}(t) = (1.0 - \alpha) \cdot \boldsymbol{P}(t) + \alpha \cdot \boldsymbol{P}_{BSF} \tag{11}$$

Given an individual X, we denote its corresponding fitness value by $f(X)$. The procedure of the new probability vector update scheme at generation t is shown in Fig.2. Note that this new update scheme generalizes the update scheme in standard PBIL. When H = 1, it is equivalent to a standard PBIL where only one best so far individual, i.e. $\boldsymbol{B}_1$, is maintained in S_{BSF}.

3.2 The Use of All-one Vector

As the problem concerned is highly constrained, $\boldsymbol{P}(t)$ in the initialization of PBIL may not be able to create feasible individuals, and thus deteriorates the effectiveness and

efficiency of PBIL. Kim *et al* [4,5] significantly improved the performance of their GA by inserting an all-one vector into the randomly generated population to enable that all merging nodes are active (coding nodes), and their GA begins with at least one feasible solution in the population.

Inspired by the above idea, we employ all-one vector(s) in the probability vector update scheme to improve the performance of the proposed PBIL. The all-one vector compensates for the absence of feasible individuals in S_{BSF} in the initialization of our algorithm. For example, if there are u ($0 < u \leq$ H) infeasible individuals in S_{BSF}, these individuals are replaced by u all-one vectors. Note that, different from the problem concerned in [4,5], our problem also considers the delay constraint. Therefore, all-one vector may still be infeasible as the delay constraint may not be met.

3.3 The Structure of the Proposed PBIL

The proposed PBIL	
1)	**Initialization**
2)	Set $t = 0$;
3)	For $i = 1, 2, \ldots, L$, set $P_i^t = 0.5$
4)	Generate a sampling set $S(t)$ of N individuals from $\boldsymbol{P}(t)$
5)	Generate a set S_{BSF} of H individuals by sampling $\boldsymbol{P}(t)$
6)	Replace infeasible individuals in S_{BSF} by all-one vectors
7)	**repeat**
8)	Set $t = t + 1$
9)	Evaluate the samples in $S(t\text{-}1)$
10)	Update $\boldsymbol{P}(t)$ by using the probability vector update scheme (Fig.2)
11)	Mutate $\boldsymbol{P}(t)$ by Eq.(9)
12)	Generate a set $S(t)$ of N samples by $\boldsymbol{P}(t)$
13)	**until** *termination condition is met*

Fig. 3. Procedure of the proposed PBIL

We use the graph decomposition method proposed in [4,5] to transform the given network G into a secondary graph G_D. In PBIL, each individual X corresponds to a secondary graph G_D. Each bit of X is associated with one of the newly introduced links between the so-called auxiliary nodes in G_D. Bit '1' and '0' means its corresponding link exists and does not exist in the secondary graph G_D, respectively.

Fitness evaluation measures each obtained G_D. For an individual X, we first check if a feasible NCM subgraph $G_{NCM}(s,T)$ can be found from its corresponding G_D. For each sink $t_k \in T$, we use the Goldberg algorithm[11], a classical max-flow algorithm, to compute the max-flow between the source s and t_k in the corresponding G_D. If all d max-flows are at least R, for each sink t_k we select R least-delay paths from all link-disjoint paths obtained from s to t_k. All the selected paths are mapped to G_D to form the $G_{NCM}(s, T)$. If the maximal path delay in the $G_{NCM}(s,T)$ satisfies the delay constraint, i.e. $D(G_{NCM}(s,T)) \leq \Omega$, we set the number of coding links in $G_{NCM}(s,T)$ to $f(X)$. If $G_{NCM}(s,T)$ cannot be found or it violates the delay constraint, X is infeasible and we set a very large fitness value Ψ to $f(X)$ (in this paper, $\Psi = 50$).

The procedure of the proposed PBIL is shown in Fig.3. The termination criteria are subject to two conditions: 1) a coding-free subgraph is obtained, or 2) the algorithm reaches a pre-defined number of generations.

4 Numerical Experiments and Discussions

We evaluate the performance of the following two algorithms:

- GA: the simple GA with binary link state encoding, tournament selection, uniform crossover, simple mutation and all-one vector inserted into initial population [5].
- PBIL-H(*num*): the proposed PBIL with the new probability vector update scheme and the use of all-one vector(s), where *num* is the integer number set to H.

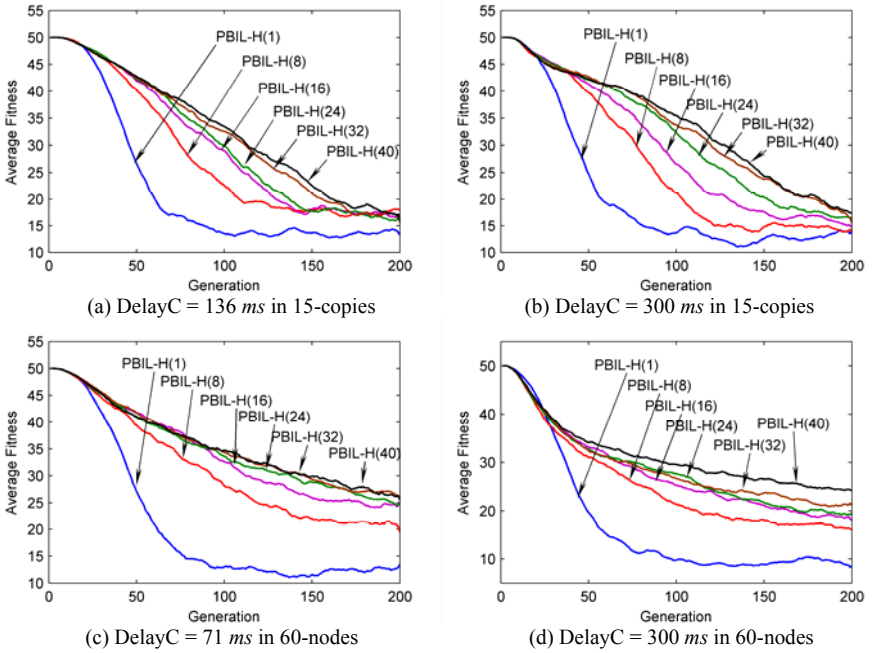
Simulations have been carried out upon two networks, i.e. a 15-copies network (with $R = 2$) adopted in [5] and a random directed network (with 60 nodes, 150 links, 11 sinks, and $R = 5$). The propagation delay over each link $D(e)$ is uniformly distributed from 2*ms* to 10*ms* and the time consumed for coding Δ_c is set to 2*ms*. For each topology, we set two different delay constraints, a severe one and a loose one, i.e. for the 15-copies topology, 136*ms* and 300*ms*, respectively; for the 60-nodes topology 71*ms* and 300*ms*, respectively. The solution lengths in 15-copies and 60-nodes are 176 bits and 235 bits, respectively. In all scenarios, the population size and the pre-defined termination generation is set to 40 and 300, respectively. In GA, tournament size, crossover and mutation probabilities are set to 2, 0.8 and 0.01, respectively. In PBIL-H(*num*), $\alpha = 0.1$, $pm = 0.02$, and $\sigma = 0.05$. To study how H affects the performance of our PBIL, we set 1, 8, 16, 24, 32, and 40 to *num*, respectively. All experimental results were collected by running each algorithm 20 times. The performance analysis is based on the following criteria:

- The evolution of the average fitness. Note that the termination condition here is that algorithm stops after 200 generations.
- The successful ratio (SR) of finding a coding-free subgraph (i.e. subgraph without coding performed). Note that the optimal solutions in our experiments are solutions that produce coding-free subgraphs.
- The average best fitness (ABF). It is the average value of the obtained best fitness values in 20 runs.
- The average computational time (ACT) to run an algorithm.

Table 1 shows the results of GA and PBIL-H(1) on the 15-copies and 60-nodes networks. Obviously, GA is beaten by PBIL-H(1) in every case, showing to be ineffective in solving the problems concerned. In terms of SR, the maximum value GA obtains is 60% while PBIL-H(1) has at least 80%. In particular, for 15-copies network, GA even cannot reach coding-free subgraph(s). In contrast, the performance of PBIL-H(1) is stabilized and with high-quality, which shows PBIL-H(1) is more effective compared with GA. We also see that PBIL-H(1) performs better than GA with respect to ABF and sometimes the superiority is substantial, e.g. in severely constrained cases.

Table 1. Results by GA and PBIL-H(1)

Evaluation Criteria		15-copies network		60-nodes network	
		Ω = *136 ms*	Ω = *300 ms*	Ω = *71 ms*	Ω = *300 ms*
SR (%)	GA	0.0	0.0	5.0	60.0
	PBIL-H(1)	100.0	95.0	80.0	80.0
ABF	GA	50.0	9.05	47.5	1.00
	PBIL-H(1)	0.0	0.05	0.20	0.20



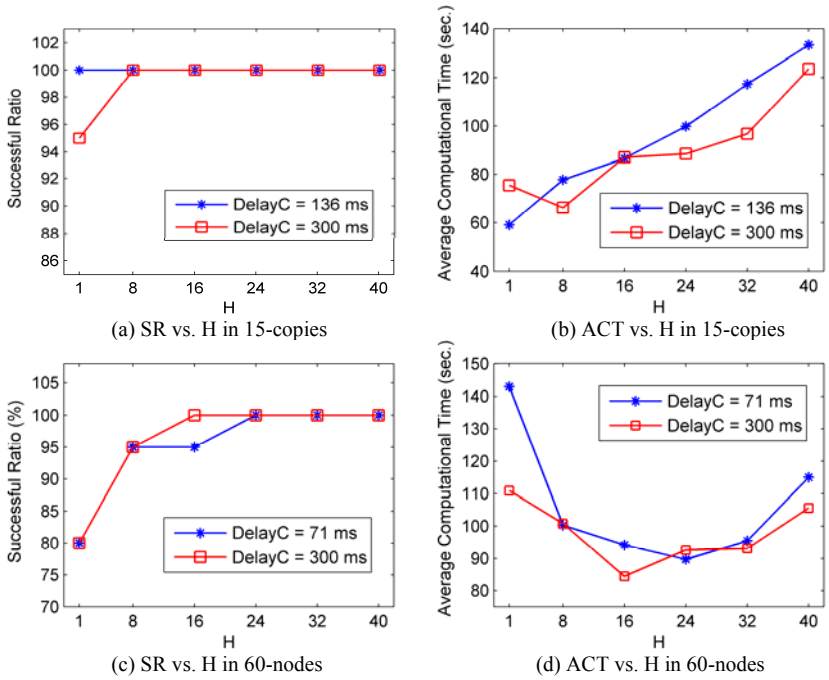
(a) DelayC = 136 *ms* in 15-copies (b) DelayC = 300 *ms* in 15-copies

(c) DelayC = 71 *ms* in 60-nodes (d) DelayC = 300 *ms* in 60-nodes

Fig. 4. Average fitness vs. generation. DelayC denotes the delay bound.

Fig.4 compares the average fitness obtained by PBIL-H(*num*) for the 15-copies and 60-nodes networks. We find that the convergence characteristics of these algorithms are affected by H, i.e. the number of best so far individuals kept in S_{BSF}, in such a way that the larger H, the slower convergence. As aforementioned, the statistical information of S_{BSF} is used to update the probability vector. With more individuals in S_{BSF}, the contribution to adjust the probability vector from a single individual in S_{BSF} becomes less. Therefore, a larger H is more likely to slow down the convergence of the proposed algorithm. However, PBILs with a smaller H may suffer from rapid diversity loss and converge to local optima.

On the same network with different delay bounds, the performances of PBIL-H(*num*) deteriorate on severely constrained cases. For example, PBIL-H(1) for the 60-nodes network with smaller delay bound in Fig.4(c) obtains a worse average fitness than that of the same network with larger delay bound in Fig.4(d). Obviously, this is due to that there are less feasible solutions in the solution space of the severely constrained topologies compared to that of loosely constrained cases.



(a) SR vs. H in 15-copies (b) ACT vs. H in 15-copies

(c) SR vs. H in 60-nodes (d) ACT vs. H in 60-nodes

Fig. 5. The effect of H

Fig.5 shows the effect of H to SR and ACT in the 15-copies and 60-nodes networks. In Fig.5(a) and (b), we notice that the larger the value of H, the higher the SR, and the larger ACT. In the case of severe delay constraint, the ACT of PBIL-H(*num*), *num* = 1, 8, 16, 24, 32 and 40, become increasingly worse. This phenomenon demonstrates the tradeoff between diversification and intensification. The more the best so far individuals are maintained in S_{BSF}, the better the diversity is kept, thus avoiding pre-maturity. However, larger H, on the other hand, slows down the convergence, where PBIL cannot efficiently exploit and find an optimal solution from a certain area of the solution space yet consumes a large amount of computational time. Concerning SR in Fig.5(c), we also notice that the larger the value of H, the higher the SR in cases with different delay constraints. On the other hand, in Fig.5(d), the ACT falls first and then rises up. This is because at beginning with H increasing, the global exploration ability of the algorithm is improved and the ACT to obtain optimal solutions is reduced. However, with H continuing to grow larger, the convergence speed of PBIL decreases, requiring more ACT before stop. If we focus on the sections of curves where SR is 100% (i.e. $H \geq 16$ in loosely constrained case and $H \geq 24$ in severely constrained case, respectively), it is also clear that larger H still results in larger ACT. From the above analysis, we find that if the value of H is properly selected, e.g. set H = 8 in 15-copies network and H = 24 in 60-nodes network, rapid convergence and global search capability can be achieved simultaneously.

5 Conclusions

In this paper, we investigate for the first time the delay constrained minimization problem on network coding operations. A population based incremental learning algorithm with a new probability vector update scheme and all-one vector(s) is proposed. The new update scheme makes use of a group of best so far individuals to adjust the probability vector. The number of best individuals in this scheme needs to be carefully devised so that high performance and low computational time are achieved at the same time. Besides, all-one vectors are adopted to drive the probability vector towards feasible solutions at the early evolutionary generations, which greatly improve the performance of our proposed algorithm. Simulation results demonstrate that the proposed algorithm is highly effective in solving the problem concerned.

Acknowledgements

This research is supported by China Scholarship Council and The University of Nottingham, UK.

References

1. Ahlswede, R., Cai, N., Li, S.Y.R., Yeung R.W.: Network coding 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. Kim, M., Ahn, C.W., Médard, M., Effros, M.: On minimizing network coding resources: an evolutionary approach. In: Proc. NetCod. (2006)
4. Kim, M., Médard, M., Aggarwal, V., O'Reilly, V., Kim, W., Ahn, C.W., Effros, M.: Evolutionary approaches to minimizing network coding resources. In: Proc. Inforcom. (2007)
5. Kim, M., Aggarwal, V., O'Reilly, V., Médard, M., Kim, W.: Genetic representations for evolutionary minimization of network coding resources. In: Proc. EvoCOMNET. (2007)
6. Fragouli, C., Soljanin, E.: Information flow decomposition for network coding. IEEE Trans. Inform. Theory **52**(3) (2006) 829-848
7. Langberg, M., Sprintson, A., Bruck, J.: The encoding complexity of network coding. IEEE Trans. Inform. Theory **52**(6) (2006) 2386-2397
8. Bhattad, K., Ratnakar, N., Koetter, R., Narayanan, K.R.: Minimal network coding for multicast. In: Proc. ISIT2005. (2005)
9. Baluja, S.: 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)
10. Larranaga, P., Lozano, J.A.: Estimation of distribution algorithms: a new tool for evolutionary computation. MA: Kluwer, Norwell (2002)
11. Goldberg, V.: A new max-flow algorithm. Technical Report MIT/LCS/TM-291, MIT (1985)