22nd International Symposium on
Mathematical Theory of Networks and Systems
July 11-15, 2016. Minneapolis, MN, USA

Combining Greedy Hyperbolic Routing with Back-pressure Scheduling for Better Network Performance

Ladan Rabieekenari and John S. Baras

Abstract— **The back-pressure algorithm (BP) has received much attention for jointly routing and scheduling over multi-hop networks. The back-pressure algorithm is throughput-optimal, but it has poor delay performance. This is due to selection of unnecessarily long paths and routing loops, as the algorithm operates without accounting for the network topology. In this paper, we propose a throughput-optimal routing and scheduling algorithm that improves delay performance by greedy embedding of the network in hyperbolic space. We improve delay performance by solving an optimization problem, which aims to send packets mostly to greedy neighbors, subject to throughput-optimality constraints. The algorithm that solves this optimization problem has a design parameter M. We study the effect of M on delay performance analytically. We validate our theoretical results via simulations and demonstrate that the proposed algorithm improves the delay performance.**

I. INTRODUCTION

In recent decades, there has been huge improvement in wireless network technology which led to new wireless networks such as sensor networks, cellular networks, mobile ad-hoc networks. In these scenarios, unlike wired networks, users compete for accessing a shared transmission medium. As a result, designing high-performance and efficient schemes for resource allocation is of great importance for such networks. Some of the metrics critical to the performance of these networks are throughput and delay. There is an increasing demand for high throughput and low delay scheduling and routing algorithms in both wireless and wired networks. High throughput is critical to respond to increasing demand of different applications. Besides that, delay is very important in real-time applications such as VoIP. Tassiulas and Ephremides in [1] proposed the back-pressure algorithm for scheduling and routing, and proved its throughput-optimality. A routing/scheduling algorithm is throughput-optimal in the sense of [1], if it can stabilize any traffic that can be stabilized by any other routing/scheduling algorithm.

The back-pressure algorithm is a congestion based routing and scheduling protocol that sends packets along the links with higher queue differential backlog. However, it has poor delay performance because it explores all feasible paths between each source and destination without considering the delay metric. This extensive hop-by-hop path exploration for each packet leads to network stability. However, this may lead to unnecessarily long paths and routing-loops. As a result, the back-pressure algorithm has poor end-to-end delay performance.

The authors are with Institute for Systems Research, and the Department of Electrical and Computer Engineering at the University of Maryland, College Park, MD 20742 {rabiee, baras}@umd.edu

There have been several studies on delay improvement in back-pressure. In [2], Ji et al. used the age of head-of-line packets instead of queue length as link weights, which is throughput-optimal for fixed routing. In [3], Ying et al. proposed an algorithm that adaptively selects a set of optimal routes between each source and destination, but increases the computational complexity and the number of queues per node considerably. Both [4] and [5] use shadow queues to improve delay performance and decrease the number of queues in the network. Both algorithms are throughput-optimal scheduling for fixed routing.

Stai et al. in [6] assigned virtual coordinates in hyperbolic space to each node such that there is a greedy path in hyperbolic space between each pair of nodes. They applied back-pressure scheduling over a fixed set of greedy paths, and called it Greedy back-pressure (GBP). In greedy back-pressure, delay performance in light loads is improved by restricting the packets to be sent along specific loop-free paths. However, it is at the cost of decreasing the capacity region, which is the main characteristic of the back-pressure algorithm. Besides that, as our results show, the delay of greedy back-pressure (as implemented in [6]) in heavy loads may be larger than traditional back-pressure. This is due to the fact that in the greedy back-pressure of [6], they restrict packets to be sent along specific paths without considering the arrival rate information, which may lead to higher congestion.

In this paper, we propose a routing and scheduling algorithm that improves delay in the back-pressure algorithm considerably, while maintaining throughput optimality. We embed the graph in hyperbolic space such that there is a greedy path between each pair of nodes. This ensures existence of greedy loop-free paths between each pair of nodes. We utilize the technique proposed in [7] to embed the network graph in hyperbolic space.

Our algorithm is the solution to an optimization problem that aims to minimize routing packets (i.e. minimize the amount of traffic routed) over non-greedy paths subject to throughput optimality constraints. The selection of the objective function ensures that packets are mostly routed through loop-free greedy paths. Since the back-pressure algorithm sends packets through routing loops, especially in light to moderate traffic, this objective function would improve delay considerably. As we demonstrate, the solution of the problem considers just greedy paths in light traffic, but as congestion in the network increases it utilizes other feasible paths in order to keep the queues in the network stable.

The rest of this paper is organized as follows. In Section II we summarize the properties of hyperbolic embedding and greedy routing. In Section III, we describe our system and state our assumptions. In Section IV we formalize our optimization problem and obtain a greedy-aided back-pressure algorithm. We study the influence of the design parameter on performance of the proposed algorithm. In Section V, we discuss complexity and distributivity of the proposed algorithm. In Section VI, we describe our simulation settings and compare the proposed algorithm with the traditional back-pressure and the greedy back-pressure. We close with some conclusions and future work in Section VII.

II. Hyperbolic embedding and greedy geographical paths

There are several different models for constructing hyperbolic geometry. One of the standard models is the Poincaré disc model. In this model, hyperbolic space $\mathbb{H}$ is represented by a set of points $(x, y) \in \mathbb{R}^2$ such that $x^2 + y^2 \leq 1$, which represents a unit disc. We refer to points in the hyperbolic plane using complex coordinates, such that (x, y) is represented by the complex number $z = x + yi$.

The hyperbolic plane has a boundary circle denoted by $\partial\mathbb{H}$, which is $x^2+y^2 = 1$, and represents the infinity. Points on $\partial\mathbb{H}$ are at infinite distance from any point inside the circle. If u and v are two points in the unit disc, the distance between these two points in the Poincaré disk model is :

$$cosh\ d_H(u, v) = \frac{2|u - v|^2}{(1 - |u|^2)(1 - |v|^2)} + 1 \qquad (1)$$

***Definition** 1:* An embedding of a graph G in $\mathbb{H}^d$ is a mapping $C(G) : V \to \mathbb{H}^d$ that assigns to each vertex $v \in V$, a virtual coordinate $C(v)$.

In greedy geographical routing, nodes forward the packets based on the coordinates of the destination and coordinates of their neighbors. Each node sends the packet to the destination by forwarding the packet to any neighbor which is closer to the destination than the node itself. If we use Euclidean coordinates of the physical location of nodes, packets may get stuck in local minima of the distance-to-destination function.

Greedy embedding is a graph embedding that makes simple greedy geometric packet forwarding successful for every source-destination pair. In [7], the authors proposed a distributed algorithm that assigns a virtual coordinate in hyperbolic plane to each node in the network, such that there exists a greedy geographical path for each pair of source and destination with respect to these virtual coordinates. To embed the actual graph G in hyperbolic plane, first an arbitrary spanning tree T of G is chosen. If T admits a greedy embedding in the hyperbolic space then G also admits the greedy embedding. So, the algorithm embeds the spanning tree in the hyperbolic plane such that tree edges provide a greedy path between each source destination pair. As a result in this embedding each node has at least one greedy neighbor to the destination, which is one of its children or its parent. We encourage the reader to read [7] for details of the algorithm.

III. System model and capacity region

Consider a network represented by a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of directed links. Nodes are wireless transmitters/receivers and links represent the wireless channel between two nodes if they can directly communicate with each other. $\mu_{m,n}$ is the maximum transmission rate supported on a directed link from node m to node n. We assume that time is slotted, with a typical time slot denoted by t. We denote $\mu_{ij}^d(t)$ as communication traffic in link (i, j) for destination d at time t. Denote by $A_i^d(t)$ the amount of new exogenous data that arrives at node i on slot t that must eventually be delivered to node d. We assume each $A_i^d(t)$ satisfies the Strong Law of Large Numbers (SLLN). That is with probability 1 we have:

$$\lim_{t\to\infty} \frac{\sum_{\tau=0}^{t-1} A_i^d(\tau)}{t} = \lambda_i^d.$$

We assume $\lambda_i^d = 0$ if $i = d$. $q_i^d(t)$ denotes the queue length of a FIFO queue at node i for destination d. A scheduling policy is a set of links that are active at the same time. A scheduling policy is called feasible if activated links do not interfere with each other. We call Γ the set of all feasible schedules. Also we use the notation $\mathcal{N}(i)$ to denote the one-hop neighbors of node i.

The capacity region of the network is defined as the set of all end-to-end traffic load matrices that can be stably supported under some network control policy. By stability we mean the time average queue length of all queues in the network doesn't go to infinity. A network policy is called throughput-optimal if its capacity region is the same as the network capacity region. In [1], the authors proved that the back-pressure algorithm is throughput-optimal for the capacity region of the network denoted as Λ_G. Λ_G is the set of all input rate matrices (λ_i^d) such that there exists a rate matrix $[\mu_{ij}]$ satisfying the following constraints:

- Efficiency constraints: $\mu_{ij}^d \geq 0,\ \mu_{ii}^d = 0,$ $\mu_{dj}^d = 0,\ \sum_d \mu_{ij}^d \leq \mu_{ij},\ \forall i, d, j.$
- Flow constraints: $\lambda_i^d + \sum_l \mu_{li}^d \leq \sum_l \mu_{il}^d,\ \forall i, d : i \neq d.$

IV. greedy-aided back-pressure

We are interested in a routing algorithm which minimizes average delay in the network subject to the throughput optimality constraint.

Assume we have hyperbolic coordinates of nodes in our network obtained through a distributed hyperbolic greedy embedding algorithm by choosing a random spanning tree. We assume our network topology does not change frequently such that at each time slot the virtual coordinates result in greedy paths between each pair of nodes.

In greedy routing, each node knows the virtual coordinates of itself, its neighbors and destination. In this routing, packets are forwarded to a neighbor which is closer to the destination than the node itself, so the distance to the destination is decreasing. Decreasing distance to the destination ensures one node cannot be passed twice, so the path is loop free. As a result if packets are sent through greedy links, they get to the destination through loop free paths. Thus, sending packets through greedy links results in low delay when the network is not congested. However, restricting packets to be sent over a fixed set of paths may result in large delay and unstable queues in heavy loads.

In order to ensure throughput optimality of the algorithm, the packets should not be restricted to go through a set of pre-specified paths for all set of packet arrivals. In order to keep the throughput optimality feature while providing good delay performance, we introduce a penalty function which is the total amount of resources used over non-greedy links. We are interested to find the routes for flows such that time average expected penalty is minimized subject to throughput optimality constraints. Thus we formulate the following optimization problem:

$$\min_{{\mu'}^{d}_{i,j}(t)} \quad \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \sum_{i,j,d\in \text{P}} \mathbb{E}\{\mu_{ij}^d(t)\} \qquad (2)$$
$$s.t. \quad \{\mu_{nj}(t)\}_{(n,j)\in L} \in \Lambda_G,$$

where P denotes the set of (i,j,d) such that node j is not a greedy neighbor of node i for destination d.
This optimization problem minimizes the total amount of resources used by non-greedy links subject to throughput optimality constraints. The solution to this problem will route packets through greedy paths unless greedy paths will lead to instability of queues.

Theorem 1: The scheduling and routing algorithm described in Algorithm 1 asymptotically solves the described optimization problem.

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{i,j,d} \mathbb{E}[{\mu'}^{d}_{i,j}(t)] = c^{opt} + O(\frac{1}{M}) \qquad (3)$$

where c^{opt} is the infimum time average cost achievable by any policy that meets the required constraints and $\mathbb{E}\left[{\mu'}^{d}_{i,j}(t)\right]$ is the average link rate when using Algorithm 1 with fixed parameter M.
Furthermore, the queues are stable and the expected value of the queue length is bounded as follows:

$$\sum_{i,d} \mathbb{E}(q_i^d[\infty]) = O(M(c^{opt} - c_{min})) \qquad (4)$$

where $\mathbb{E}[q_i^d(\infty)]$ is the queue length as $t \to \infty$ and c_{min} represents a lower bound of the cost function.

Proof: We denote by $Q(t) = (q_i^d(t))$, the matrix of queues in the network. We define the Lyapunov function $L(Q(t)) = \sum_{i,d} q_i^d(t)^2$. Based on chapter 4 of [8], we need to design a controller that, at every time slot t, observes the $Q(t)$ values and subject to the known $Q(t)$ greedily minimizes the drift-plus-penalty expression which is as follows:

$$\mathbb{E}\{L(Q(t+1)) - L(Q(t)) \mid Q(t)\} + M\mathbb{E}\{\sum_{i,j,d\in \text{P}} \mu_{ij}^d(t) \mid Q(t)\} \qquad (5)$$

where $M > 0$ is a control parameter that affects performance-delay trade-off.
Intuitively, minimizing $\mathbb{E}\{L(Q(t+1)) - L(Q(t)) \mid Q(t)\}$ alone would tend to push the network to a lower congestion state, however, it may result in large penalty. Thus, we minimize a weighted drift-plus-penalty, where M represents how much we emphasize penalty minimization.
In order to minimize the drift-plus-penalty expression, we define two indicator functions:
$I_1(i,j,d) = \{dist_H(i,d) > dist_H(j,d) \wedge (j \in \mathcal{N}(i))\}$ which means link (i,j) is a greedy path for destination d.
$I_2(i,j,d) = \{dist_H(i,d) < dist_H(j,d) \wedge (j \in \mathcal{N}(i))\}$ which means link (i,j) is not a greedy path for destination d.

We observe that $I_1 \cap I_2 = \varnothing$. So we have:

$$\sum_{i,j,d} \mu_{ij}^d = \sum_{(i,j,d)|I_1} \mu_{ij}^d + \sum_{(i,j,d)|I_2} \mu_{ij}^d \qquad (6)$$

The queue dynamics are:

$$q_i^d(t+1) = \max\{q_i^d(t) - \sum_j \mu_{ij}^d(t), 0\} + \sum_j \mu_{ji}^d(t) + A_i^d(t) \qquad (7)$$

Based on Lemma 4.3 [9], if V, U, μ, A are all non-negative numbers and $V \le \max[U-\mu, 0] + A$ then the following holds:

$$V^2 \le U^2 + \mu^2 + A^2 - 2U(\mu - A) \qquad (8)$$

We derive an upper-bound of the drift-plus-penalty expression as follows:

$$\begin{aligned}
&\mathbb{E}[L(Q(t+1)) - L(Q(t))|Q(t)] + M\sum_{(i,j,d)|I_2} \mathbb{E}[\mu_{ij}^d(t)|Q(t)] \\
&= \mathbb{E}\left[\sum_{i,d} q_i^d(t+1)^2 - \sum_{i,d} q_i^d(t)^2 \middle| Q(t)\right] \\
&+ M\sum_{(i,j,d)|I_2} \mathbb{E}[\mu_{ij}^d(t)|Q(t)] \le^{(7),(8)} \sum_{i,d}\Big\{ q_i^d(t)^2 \\
&+ \mathbb{E}\big[(\sum_j \mu_{ij}^d(t))^2 + (\sum_j \mu_{ji}^d(t) + A_i^d(t))^2 |Q(t)\big]\Big\} \\
&- \sum_{i,d}\Big\{ 2q_i^d(t)\mathbb{E}\big[(\sum_j \mu_{ij}^d(t) - \sum_j \mu_{ji}^d(t) - A_i^d(t))|Q(t)\big]\Big\} \\
&- \sum_{i,d} q_i^d(t)^2 + M\sum_{(i,j,d)|I_2} \mathbb{E}[\mu_{ij}^d(t)|Q(t)] \\
&\le B + 2\sum_{i,d} q_i^d(t)\lambda_i^d + M\sum_{(i,j,d)|I_2} \mathbb{E}[\mu_{ij}^d(t)|Q(t)] \\
&- 2\sum_{i,d} q_i^d(t)\mathbb{E}\left[\left(\sum_j \mu_{ij}^d(t) - \sum_j \mu_{ji}^d(t)\right)\middle| Q(t)\right] \qquad (9)
\end{aligned}$$

where B is an upper-bound of

$$\sum_{i,d}\mathbb{E}\left[\left(\sum_j \mu_{ij}^d(t)\right)^2 + \left(\sum_j \mu_{ji}^d(t) + A_i^d(t)\right)^2 \middle| Q(t)\right]$$

As a result by applying (6) in (9) we have

$$\mathbb{E}(L(Q(t+1)) - L(Q(t))|Q(t)) + M \sum_{(i,j,d)|I_2} \mathbb{E}[\mu_{ij}^d(t)|Q(t)]$$
$$\leq B + 2\sum_{i,d} q_i^d(t)\lambda_i^d - 2\sum_{i,d}\sum_{(i,j)|I_1}\Bigg\{\Big(q_i^d(t) - q_j^d(t)\Big)$$
$$\mathbb{E}\Big[\mu_{ij}^d(t)\Big|Q(t)\Big]\Bigg\} - 2\sum_{i,d}\sum_{(i,j)|I_2}\Bigg\{\Big(q_i^d(t) - q_j^d(t)\Big)$$
$$\mathbb{E}\Big[\mu_{ij}^d(t)\Big|Q(t)\Big]\Bigg\} + M \sum_{(i,j,d)|I_2} \mathbb{E}[\mu_{ij}^d(t)|Q(t)]$$
$$= B + 2\sum_{i,d} q_i^d(t)\lambda_i^d - 2\sum_{i}\Bigg\{\sum_{j,d|I_2}(q_i^d(t) - q_j^d(t) - \frac{M}{2})$$
$$\mathbb{E}[\mu_{ij}^d(t)|Q(t)] + \sum_{j,d|I_1}(q_i^d(t) - q_j^d(t))\mathbb{E}[\mu_{ij}^d(t)|Q(t)]\Bigg\} \quad (10)$$

Every timeslot, the control decision variables are chosen to minimize the right hand side of the above inequality which results in Algorithm 1 (substitute $\frac{M}{2}$ with M). ■

Algorithm 1 Greedy-aided back-pressure (GA-BP)

```
 1: ▷ Each node $i$ maintains a separate queue for each destination $d$
 2: for each directed link $(i,j)$ do
 3:     for each destination d do
 4:         ▷ if node $j$ is a greedy neighbor of node $i$
 5:         if $dist_H(i,d) > dist_H(j,d)$ then
 6:             $P_{ij}^d(t) \leftarrow q_i^d(t) - q_j^d(t)$
 7:         else
 8:             $P_{ij}^d(t) \leftarrow q_i^d(t) - q_j^d(t) - M$
 9:         end if
10:     end for
11:     ▷ Each link is assigned a weight $P_{ij}$
12:     $P_{ij}(t) \leftarrow \max\{\max_d P_{ij}^d(t), 0\}$;
13:     ▷ The destination which achieves the maximum in previous line
14:     $d^*(i,j,t) \leftarrow \arg\max_d P_{ij}^d(t)$;
15: end for
16: ▷ Scheduling and routing rule: Choose the rate matrix through the maximization:
17: $[\mu_{ij}(t)] \leftarrow \arg\max_{\mu' \in \Gamma} \sum_{(i,j)} \mu'_{ij} P_{ij}(t)$
18: for each directed link $(i,j)$ do
19:     if $\mu_{ij}(t) > 0$ then
20:         the link $(i,j)$ serves $d^*(i,j,t)$ with $\mu_{ij}^{d^*}(t) = \mu_{ij}(t)$
21:     end if
22: end for
```

In Algorithm 1, $M \geq 0$ is a design parameter. The algorithm prioritizes routing packets along greedy paths over non-greedy ones in order to improve delay performance while achieving throughput-optimality. With this change, one node can send packets to the non-greedy neighbor if and only if the queue differential backlog between the node and the non-greedy neighbor exceeds M. If we choose $M = 0$, GA-BP would be the same as the traditional BP. Since hyperbolic embedding in [7] guarantees existence of a greedy path to the destination, the routing algorithm proposed in Algorithm 1 ensures packets would be routed to the destination.

As shown in Theorem 1, the performance of the algorithm which solves the optimization problem with fixed parameter M is within $O(\frac{1}{M})$ of the optimal solution. However, the total queue length in the network increases linearly with M (or, equivalently, delay by Little's law). Very large M results in larger congestion while too small M results in being far from the optimal solution. So we are interested in M which is neither too large nor too small. The intuition is that greedy paths don't always provide best set of paths. By choosing small M we will prevent large delay in light loads by pushing packets to go through loop-free paths while as congestion in the network increases the greedy paths may not guarantee stability of queues. Then the packets would be sent through non-greedy paths besides greedy ones.

V. Complexity and Distributivity of Greedy-aided back-pressure

In this section, we compare the computational complexity and distributivity of the proposed algorithm to the traditional back-pressure algorithm. Intuitively, it may be observed that the complexity of our method is similar to the traditional back-pressure. In the first stage of our method, we embed the graph in hyperbolic space by assigning a hyperbolic coordinate to each node. This involves the overhead of computing a spanning tree in the graph. However, since this step is only required once in the beginning, the overhead may be negligible over the lifetime of the network. In both algorithms, each node computes and utilizes queue differential backlogs (with a bias factor for our algorithm) to make routing decisions. Thus, the recurring steps in each algorithm have the same complexity.

Even though the traditional back-pressure is a centralized algorithm, several studies on suitable distributed scheduling algorithms that achieve throughput optimality have been proposed in the literature, e.g. [10]-[11]. We may modify the weights assigned to links in these scheduling algorithms by utilizing the metrics from GA-BP. Thus, these algorithms can be utilized to substitute the scheduling in GA-BP, enabling the execution of our algorithm in a distributed manner while satisfying throughput optimality. Since the hyperbolic coordinates are calculated in a distributed manner, as demonstrated in [7], a fully distributed implementation of our algorithm is possible.

VI. Simulations and Results

In this section, we evaluate the delay and throughput performance of greedy-aided back-pressure (GA-BP) and compare it to BP and GBP via simulations. We consider two networks of wireless nodes distributed over a region and a wireline network which represents the GMPLS network of North America [12]. We assume a one hop interference model between the links for wireless networks. We assume a Poisson arrival process with mean λ for each flow in the network. In the simulations, we observed the performance of the algorithms under different traffic loads and network topology. For each, the simulation is executed for 50000 iterations.

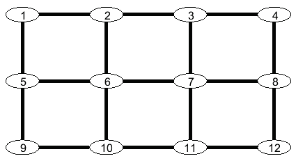


Fig. 1. Grid network topology

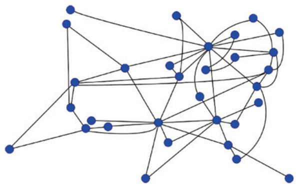


Fig. 2. Sprint GMPLS network topology of North America [12]

We consider three scenarios with varying network topology depicted in Fig 1, 2 and 3. The scenarios are selected to contrast extreme performance scenarios in the current algorithms. In the first and second scenarios, GBP has poor performance in heavy loads and it achieves only about 50% of the network capacity. In the third scenario, GBP almost achieves the capacity region of the network. We study the performance of GA-BP in all scenarios and compare it with the performance of GBP and BP.

The optimal throughput region is defined as the set of arrival rates in which queue length and thus delay remains finite. We can consider the traffic load under which the queue length and thus delay increases rapidly as the boundary of the optimal throughput region.

TABLE I
SET OF FLOWS IN GRID NETWORK

Flow ID	**(Source,Destination)**
1	(3,11)
2	(5,12)
3	(2,9)
4	(1,8)

In the first scenario, shown in Figure 1, the network has a grid structure with 12 nodes and 17 links. 4 flows are created in the network as shown in Table I. The arrival rate of each flow ranges from 0.05 to 1.3. Each link can transmit three

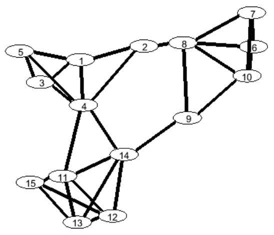


Fig. 3. Random network topology

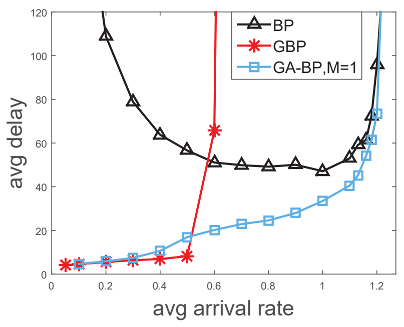
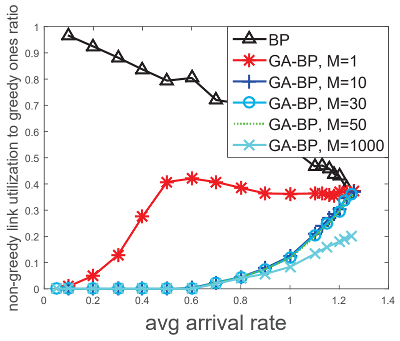
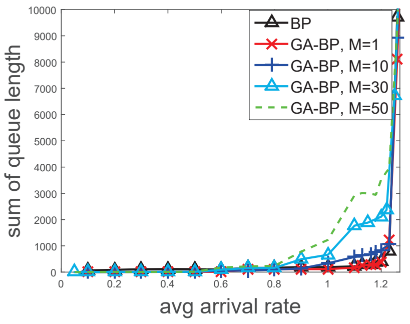


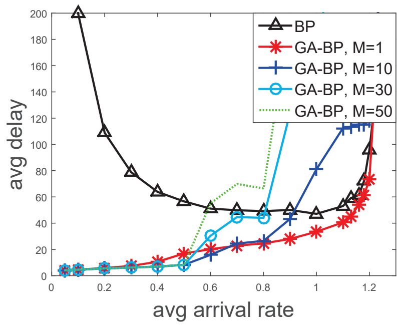
Fig. 4. Average delay vs. average arrival rate in scenario 1



(a)



(b)



(c)

Fig. 5. Performance parameters in scenario 1 for varying M: (a) ratio of total packets routed over non-greedy links to the packets routed over greedy ones; (b) sum of queue length vs. average arrival rate; (c) average delay vs. average arrival rate

packets during a time slot. We find the hyperbolic embedding of the graph for a minimum spanning tree (by assigning weight equal to 1 to each link) with 1 as the root node.

Figure 4 shows delay as a function of the arrival rate for the three algorithms BP, GBP and GA-BP with $M = 1$. It can be seen GA-BP and BP achieve the same capacity region boundary which supports our theoretical results on throughput optimality. Moreover, GA-BP achieves better delay performance compared to BP and GBP. This is because under the back-pressure algorithm, packets are sent over routing loops and unnecessarily long paths when there is not enough congestion in the network. This leads to poor delay performance especially in light and moderate traffic. By using GA-BP, packets are routed through loop free paths of the greedy embedding in light loads. However, as the gradients build up toward the destination, packets are also forwarded through non-greedy paths which decreases the congestion in the network. So GA-BP improves delay by routing packets through shorter paths, while it also exploits long paths in the heavy traffic regime.

We vary our control parameter M to study its impact on the performance of GA-BP. In Figure 5(a), we illustrate the impact of M on the ratio of total packets routed over non-greedy links to the packets routed over greedy ones. It can be seen that as M increases the ratio of the packets routed over non-greedy links over greedy ones decreases. In Theorem 1, we have proved that the average packets sent through non-greedy links are asymptotically minimized when $M \to \infty$. Our simulation results are consistent with the theorem.

In light traffic, for the case of $M = 0$ (traditional back-pressure), the ratio is large. This is because in very light loads, there is not sufficient traffic in the network. So it takes very long time to build up gradient toward the destination. As a result, the packets choose their next hop randomly. As the arrival rate increases, the gradients towards the destinations build up faster. Thus, the packets traverse mostly through short loop free paths. Since greedy paths also contain a set of short loop free paths, when $M = 0$, as arrival rate increases, the ratio decreases. As shown in Figure 5(a), the ratio of GA-BP became closer to that of the back-pressure algorithm. This is because in a heavy traffic regime, GA-BP also exploits non-greedy paths to maintain stability. As M increases, higher priority is assigned to send packets through greedy links, so as expected the ratio decreases.

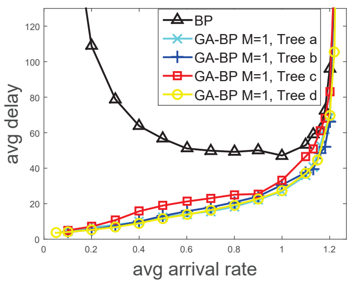


Fig. 6. Performance parameters in scenario 1 with different spanning trees

TABLE II
SET OF FLOWS IN RANDOM NETWORK

Flow ID	**(Source,Destination)**
1	(1,14)
2	(5,15)
3	(12,3)
4	(7,13)

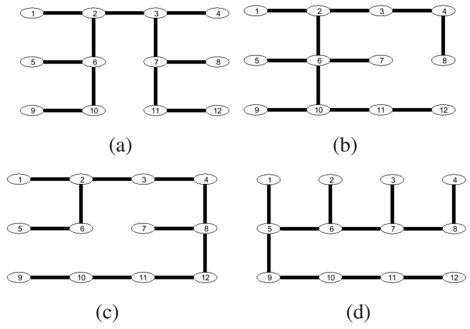


Fig. 7. Spanning trees used for embedding in grid network topology

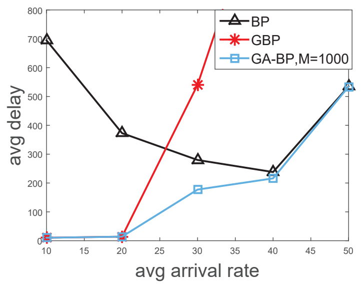
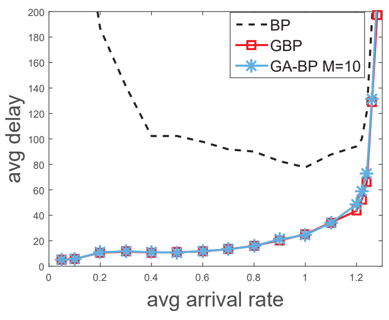


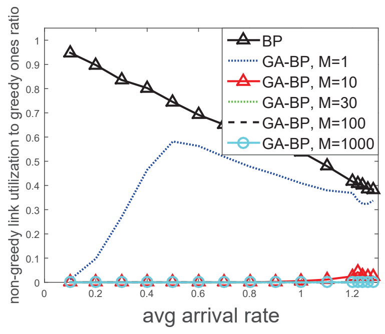
Fig. 8. Average delay vs. average arrival rate in scenario 2

Next, we study the effect of M on the sum of queue length in the network. In Figure 5(b), we illustrate congestion in the network when using GA-BP for various values of M. As stated in the Theorem 1, the upper-bound on sum of instantaneous queue length in the network increases as M increases. Our simulation results are consistent with the theorem.

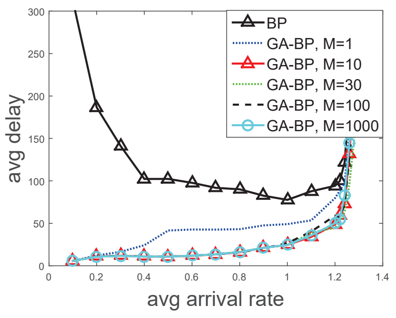
Next, we study the effect of M on delay performance. Figure 5(c) depicts the delay performance for varying values of M. The delays for different values of M (except 0) are almost the same in the light traffic region. However, in moderate to heavy traffic, small M leads to better delay performance. This is because as stated in the Theorem 1, the sum of queue lengths in the network is bounded by a term proportional to M. So for large M, total congestion in the network increases which results in larger delay. On the other hand, for very small $M (M = 0)$ the ratio of the packets sent through non-greedy links increases which causes loops and long paths in light loads. As M increases, the nodes prefer to send the packets to greedy neighbors. Thus for heavy load scenario, this significantly increases congestion along those paths, leading to an increase in delay. It can be observed that while setting $M = 50$ results in an improvement in delay for light load, it has the opposite effect for heavy load. This



(a)



(b)



(c)

Fig. 9. Performance parameters in scenario 3: (a) average delay vs. average arrival rate; (b) ratio of total packets routed over non-greedy links to the packets routed over greedy ones; (c) average delay vs. average arrival rate varying M.

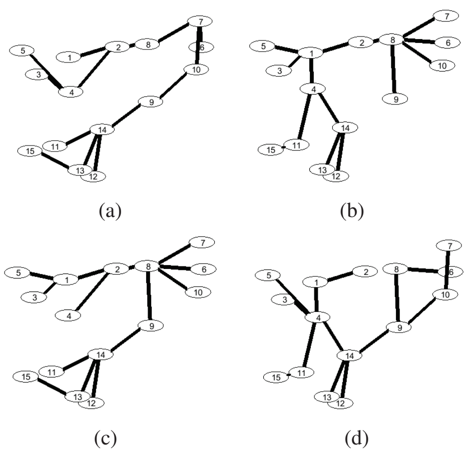


Fig. 10. Spanning trees used for embedding in random network topology

clearly highlights the influence of congestion along greedy paths as we increase the load.

Next, we study the effect of spanning tree on the performance of GA-BP. As stated earlier, embedding different spanning trees results in different set of paths between each pair of source and destination. It is important to notice, a node may have more than one greedy neighbor toward a destination. This means the greedy paths are not limited to paths of the spanning tree. We are interested in a routing algorithm which would forward packets through short paths. The long routes would only be used when the short routes are heavily loaded. As a result, we are interested in a spanning tree whose embedding results in short greedy paths. Based on the GA-BP algorithm, as congestion in greedy paths increases, the packets are sent through non-greedy paths besides greedy ones.

Figure 6 illustrates the impact of the spanning tree used for embedding on delay performance of GA-BP. The graph is embedded by the spanning trees provided in Figure 7. As Figure 6 shows, embedding of 7(c) results in larger delay compared to others. The reason is that the greedy paths obtained by embedding 7(c) do not contain short paths for all flows. The shortest greedy path for flow from node 2 to node 9 provided by embedding of 7(c) is 4 hop long, which is one hop larger than the length of the corresponding shortest path. Besides that, the average length of greedy paths provided by 7(c) is larger than others.

In the second scenario, we consider a wireline network shown in Figure 2, which represents the GMPLS network topology of North America [12]. This network has 31 nodes and 52 links. We assume each link can transmit 64 packets during a time slot. Each node at each time slot chooses a random destination and generates traffic for it with average arrival rate equal to $\lambda = 10$ to $\lambda = 50$. We construct the hyperbolic embedding of the graph for one arbitrary spanning tree.

In Figure 8, we compare the delay performance of GA-BP when $M = 1000$ with GBP and BP. It can be seen that GA-BP achieves better delay performance compared to BP and GBP.

As stated earlier, we are interested to use long routes when the short routes are congested. We can detect congestion over greedy paths and start sending packets through non-greedy links by choosing a proper M. Increasing M will

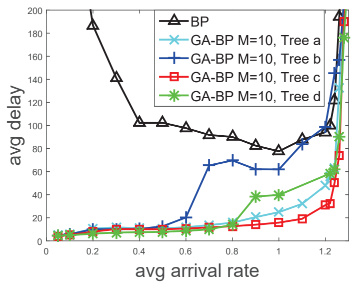


Fig. 11. Performance parameters with different spanning trees

delay sending packets over non-greedy routes. Proper choice of M depends on the average queue length when the packets are sent just through greedy paths and the network is stable. If the maximum achievable average queue length of a network is larger compared to another network, based on the characteristics of back-pressure, average queue differential backlog in this network is larger. As a result, a larger M should be chosen in order to prevent packets to be sent along non-greedy links before the greedy ones become heavily loaded. If the maximum achievable average queue length is large, small M results in sending packets through non-greedy links before congestion happens. In this scenario, the average length of active queues when $\lambda = 20$ is equal to 31. This average queue length is much larger than 2.5 which is the average queue length when $\lambda = 0.5$ in scenario 1. As a result, proper M in scenario 2 is larger than the one in scenario 1.

For the third scenario, as illustrated in Figure 3, we consider 15 nodes with 58 links. 4 flows are created in the network as shown in Table II. The arrival rate of each flow ranges from 0.05 to 1.3. We assume each link can transmit five packets during a time-slot. We construct the hyperbolic embedding of the graph for one arbitrary spanning tree.

In Figure 9(a), we compare the delay performance of GA-BP when $M = 10$ with GBP and BP. It can be seen that the throughput optimal region of GBP is the same as BP. And GA-BP and GBP perform considerably better than BP. Based on this figure, we conclude the greedy paths are performing well for this set of sources and destinations in the network. As expected in this scenario, proper M is larger than scenario 1. The reason is that the average queue length when λ is close to the capacity region, using the GBP algorithm, is about 5, which is larger than the corresponding number in scenario 1.

In Figure 9(b), we illustrate the impact of M on the ratio of total packets routed over non-greedy links to the packets routed over greedy ones. It can be seen that as M increases, the ratio of the packets routed over non-greedy links over greedy ones decreases. Since routing over the greedy paths in this scenario achieves the capacity region of the network, we can achieve throughput optimality without sending packets through non-greedy links which results in zero penalty. Based on Theorem 1, since the zero penalty is achievable, the upper bound of average sum of queue length would not depend on M. As a result, as M increases the upper bound of sum of queues remains the same. On the other hand, as M increases packets are sent through loop free greedy paths which improves delay performance.

In Figure 11, we compare the delay performance of GA-BP with $M = 10$ for different chosen spanning trees depicted in Figure 10. Greedy paths obtained by embedding 10(c) provide more short greedy paths compared to others. As a result, Tree 10(c) has better delay performance.

Simulation results in this section show that the GA-BP algorithm has better performance and it achieves the capacity region of the network. In GA-BP, M is a critical parameter that should be selected carefully. The selected M should neither be too large nor too small compared to the scale of the queue length.

VII. Conclusions and Future Work

In this paper, we have proposed a greedy-aided back-pressure algorithm to improve the delay performance, while maintaining the throughput-optimality property of the traditional back-pressure algorithm. We analyzed the proposed algorithm analytically and via simulations. We demonstrated the improvement in delay performance of our algorithm over traditional routing schemes. Our algorithm provides the network designer a control parameter M to tune the delay-performance of the network. Further, our algorithm is robust to addition of new nodes. Due to the incremental property of hyperbolic embedding, there is no need to re-embed the network. However, for real network deployments, link and node failures may occur frequently. This may lead to loss of greedy paths, causing the packets to get stuck in local minima. This requires an adaptation in the greedy route selection. We are currently investigating the dynamic network scenarios that account for such node failures.

Acknowledgment

This material is based upon work supported by the National Science Foundation (NSF) grant CNS-1035655 and NIST grant 70NANB11H148.

References

[1] L. Tassiulas and A. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,” *IEEE Trans. on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, Dec 1992.

[2] B. Ji, C. Joo, and N. Shroff, “Delay-based back-pressure scheduling in multi-hop wireless networks,” in *Proc of IEEE INFOCOM*, April 2011, pp. 2579–2587.

[3] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, “On combining shortest-path and back-pressure routing over multihop wireless networks,” *IEEE/ACM Trans. on Networking*, vol. 19, no. 3, pp. 841–854, June 2011.

[4] L. Bui, R. Srikant, and A. Stolyar, “A novel architecture for reduction of delay and queueing structure complexity in the back-pressure algorithm,” *IEEE/ACM Transactions on Networking*, vol. 19, no. 6, pp. 1597–1609, Dec 2011.

[5] B. Ji, C. Joo, and N. Shroff, “Throughput-optimal scheduling in multihop wireless networks without per-flow information,” *IEEE/ACM Trans. on Networking*, vol. 21, no. 2, pp. 634–647, April 2013.

[6] E. Stai, J. S. Baras, and S. Papavassiliou, “Throughput-delay tradeoff in wireless multi-hop networks via greedy hyperbolic embedding,” in *Proc. of Intl. Symposium on Mathematical Theory of Networks and Systems*, ser. MTNS ’12, 2012, pp. 1–8.

[7] A. Cvetkovski and M. Crovella, “Hyperbolic embedding and routing for dynamic graphs,” in *Proc. of IEEE INFOCOM*, Apr 2009, pp. 1647–1655.

[8] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan and Claypool Publishers, 2010.

[9] L. Georgiadis, M. J, and R. Tassiulas, “Resource allocation and cross-layer control in wireless networks,” in *Foundations and Trends in Networking*, 2006, pp. 1–149.

[10] A. Eryilmaz, A. Ozdaglar, and E. Modiano, “Polynomial complexity algorithms for full utilization of multi-hop wireless networks,” in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE.* IEEE, 2007, pp. 499–507.

[11] E. Modiano, D. Shah, and G. Zussman, “Maximizing throughput in wireless networks via gossiping,” *SIGMETRICS Perform. Eval. Rev.*, vol. 34, no. 1, pp. 27–38, Jun. 2006.

[12] “Sprint ip network performance,” https://www.sprint.net/performance/, 2011.