PIECEWISE RATE ALLOCATION FOR DEEP SPACE NETWORKS

Zhenghui Liu, Lixiang Liu, Jianzhou Chen
Science and Technology on Integrated Information System Laboratory
Institute of Software, Chinese Academy of Sciences
Beijing, China
{zhenghui2014, lixiang, jianzhou08}@iscas.ac.cn

ABSTRACT
To efficiently utilize link resources, rate allocation is of great importance in deep space networks. The major challenge is the incompatibility of adjacent links’ capacity, which easily leads to congestion or low link utilization. However, traditional solutions like NUM fail to capture such link variability. Therefore, we devise a piecewise model which divides a deep space network into segments according to link lengths. Then, we propose a delay-aware utility objective for each segment considering the trade-off between throughput and end-to-end delays. Correspondingly, a piecewise rate allocation algorithm with quadratic convergence rate is derived. The numerical results demonstrate that our algorithm can improve throughput by 35% and reduce end-to-end delays by 90% compared to DNUM-based algorithms.

KEY WORDS
Optimization; Deep space networks; Rate allocation; Piecewise control.

1. Introduction

With the growth of deep space exploration missions, space agencies (e.g. NASA, ESA) are committed to developing next generation of deep space networks. Different from territorial wireless networks, deep space networks have long-haul and intermittent links [1]. For example, the distance between Earth and Mars is 400 million kilometers at most and loss of line of sight occurs constantly. To build a stable communication path between two planets, multiple relay nodes are required, as shown in Figure 1. In this scenario, the network consists of different distance scales. That is the distance between node A and node B is much larger than that between node A and the earth. Consequently, it would produce incompatibility of adjacent links’ capacity during data transmission, which easily leads to congestion or low link utilization.

Therefore, it is of great importance to plan the transmission, for instance allocating rates, so as to efficiently utilize link resources. There has been a long existing interest on rate allocation [2]-[7]. As an efficient framework, network utility maximization (NUM) has been exploited to devise rate allocation strategies in many scenarios [2]-[4]. In [2], the rate allocation problem is formulated as a constrained maximization of some utility function, which is solved based on dual decompositions and sub-gradient methods. In [4], a distributed Newton-type fast converging algorithm is developed to solve NUM problems. However, NUM framework can not capture the variability of networks substantially. To overcome the defect, dynamic network utility maximization (DNUM) is proposed by N. Trichakis et al. [5]. DNUM is a multi-period version of NUM which includes delivery constraints. Then, Ermin Wei et al. develop a distributed algorithm for DNUM [6], which is based on matrix splitting techniques. In [7], a variant of DNUM is investigated to satisfy constraints on end-to-end delays. Although DNUM takes the link variability into account in multiple periods, it does not consider the case of large distance difference between adjacent links. If transmission rates keep the same, it would reduce network throughput during data transmission. Moreover, DNUM considers that network routes and link capacity are fixed in a single period and it can not satisfy optimal link utilization in different scales of the network.

Thus, in this paper, we explore the feasibility of capturing the link variability by differentiating the different scales of the network. First, we propose a piecewise model according to the difference between adjacent links, which divide the network into several segments. Then, we give the definition of dividing points and introduce storage cost between adjacent segments. For each segment, we propose a new delay-aware utility objective so as to obtain the trade-off between throughput and end-to-end delays. In addition, we devise a corresponding algorithm which has quadratic convergence rate. The numerical results show that our algorithm can
improve throughput by 35% and reduce end-to-end delays by 90% compared to DNUM-based algorithms.

The rest of this paper is organized as follows: Section 2 defines the piecewise model. Section 3 gives the definition of dividing points and the reformulation of the problem. Then, the piecewise rate allocation algorithm is derived. In Section 4, we evaluate our algorithm. The conclusion is given in the last section.

2. Problem Definition

2.1 Traditional Model

Consider a deep space backbone network. Each planet is represented by a node in the network. Assume that all planets’ orbits are nearly circular in the same plane, as shown in Figure 1. Assume that the network consists of a set \( \mathcal{V} = \{1, \ldots, V\} \) of nodes and a set \( \mathcal{L} = \{1, \ldots, L\} \) of links.

The link capacity is represented by \( c \). There is a set \( S = \{1, \ldots, S\} \) of flows. Let \( x_s \) be the transmission rate of flow \( s \) . Define \( x = [x_s]_{s \in S} \) as the vector of transmission rate of flows. The route for each flow is represented by

\[
R_s = \begin{cases} 
1 & \text{if } s\text{th flow passes through } l, \\
0 & \text{otherwise.}
\end{cases}
\]

The capacity constraint for link \( l \) is given by

\[
\sum_{s \in S} R_s x_s \leq c_l.
\]

That is, the sum of traffics through link \( l \) should be no greater than \( c_l \). Then, the problem is formulated as

\[
P : \max U \\
\text{subject to } Rx \leq c, \ x \geq 0,
\]

where \( U \) denotes the utility of the network.

By solving \( P \), we can obtain the fixed rates of flows in the network. However, the distances of adjacent links vary widely in deep space networks. It makes link capacity and routes change easily during transmission. For example, there are four nodes and one flow, as shown in Figure 2(a). Assume that the distance between node \( v_1 \) and node \( v_2 \) is large, the distance between node \( v_2 \) and node \( v_3 \) changes during the propagation from node \( v_1 \) to node \( v_2 \), as shown in Figure 2(b). If \( x_i \) remains the same, it requires more transmission power at node \( v_2 \). However, it is expensive for a deep space node with limited resources to adapt its power. A reasonable solution is to store the received data and forward at a different rate, say \( x_i \) ’ ( \( x_i \) ’ \( \leq x_i \) ). Thus, we divide the network into two segments and re-compute the rate for node \( v_2 \).

2.2 Piecewise Model

Motivated by above example, we propose a piecewise model for deep space networks. The goal is to maximize network utility with awareness of the link variability. In the model, if the difference of two adjacent links’ lengths is large, we re-compute the rates for flows. Consequently, the network is divided into several segments with separated computation.

Note that the store-and-forward method would introduce extra storage cost between adjacent segments. As we can see in Figure 2, if \( x_i \) ’ \( < x_i \), packets would be accumulated, demanding more storage at node \( v_2 \). Thus, the objective function consists of two parts: one part is the utility of each segment and the other part is the storage cost between adjacent segments.

Assume that the integral network is grouped into \( m+1 \) segments, \( S_j \ (i = 0, \ldots, m) \) denotes the node set of the \( i \)th segment, and \( M_j \) represents the \( j \)th storage cost between the \( j \)th segment and the \( (j+1) \)th segment. Then we have

\[
U = \sum_{i=0}^{m} U_{S_i} - w \sum_{j=1}^{m} M_j, \quad (1)
\]

We define \( M_j \) as

\[
M_j = \begin{cases} 
\sum_{k=0}^{(i-1)} \alpha \cdot (x_i^{(j)} - x_i^{(j+1)}) & \text{if } x_i^{(j)} > x_i^{(j+1)}, \\
0 & \text{if } x_i^{(j)} = x_i^{(j+1)},
\end{cases}
\]

where the storage cost \( M_j \) represents the accumulation of packets and it is proportional to the rate difference. \( S(j) \) denotes the flow set of the \( j \)th segment. We use \( S(j) \cap S(j+1) \) to find the common flows between two segments. \( x_i^{(j)} \) is the rate of flow \( k \) in the \( j \)th segment. \( \alpha \) is a positive coefficient.
3. Piecewise Rate Allocation

In this section, we solve the reformed problem and derive the corresponding rate allocation algorithm.

3.1 Segmentation

To divide the network, we employ a judging variable $\delta_i$ for node $i$. If the difference $\Delta$ between the lengths of links which flow into node $i$ (represented as $d_{in-i}$) and the lengths of outgoing links from node $i$ (represented as $d_{out-i}$) is large, $i$ is chosen as a dividing point in the network. We select the exponential function which is monotonically decreasing when $0<\gamma<1$ to compute $\delta_i$. Considering the scenario with multiple routes and multiple flows, the function is defined as

$$
\delta_i = \gamma^{\sigma-\lambda}, \quad \lambda = \max \left( \frac{d_{\text{max}}}{d_{\text{min}}}, \frac{d_{\text{max}}}{d_{\text{min}}} \right), \quad 0 < \gamma, \sigma < 1,
$$

where $\gamma$ and $\sigma$ are regulatory factors, $d_{\text{max}}$ represents the maximum length of ingoing links of node $i$ and $d_{\text{min}}$ denotes the minimum length of outgoing links of node $i$. Node $i$ is selected as a dividing point when $\delta_i < \Delta$, where $\Delta$ is a predefined threshold. In the experiment, we let $\Delta = 0.1$.

A. Delay-aware Utility inside a Single Segment

In order to improve throughput as well as reduce average delay in a single segment, we use a weighted trade-off between throughput and delay as the objective. In the $i$th segment, assume that the throughput utility for flow $s$ is $U_{si}^{(i)}(x_{si}^{(i)})$, $s \in \mathcal{S}(i)$, and the delay cost is $T_i$. Then, the utility equation of the $i$th network segment is given as

$$
U_{si} = (1-w) \cdot \sum_{s \in \mathcal{S}(i)} U_{si}^{(i)}(x_{si}^{(i)}) - w \cdot T_i,
$$

where $0 < w < 1$. $U_{si}^{(i)}(x_{si}^{(i)})$ is assumed to be continuous, strictly concave, monotonically increasing, and twice continuously differentiable on the feasible domain.

To obtain the average end-to-end delay in the $i$th segment, we employ a queue model based on M/M/1 [7]. Assume that the packet arrival rate $\lambda$ satisfies Poisson distribution and $\lambda_i^{(i)}$ is the rate of link $l$ in the $i$th segment, we have

$$
\lambda_i^{(i)} = \sum_{s \in \mathcal{S}(i)} R_{si}^{(i)} x_{si}^{(i)} / K,
$$

where $K$ is the packet length. Define $r_i$ as the total message arrival rate from flows in the $i$th network segment, it satisfies

$$
r_i = \sum_{s \in \mathcal{S}(i)} x_{si}^{(i)} / K.
$$

Considering transmission and queueing delay, the average cost function of delay can be defined as

$$
T_i = \frac{1}{r_i} \sum_{s \in \mathcal{S}(i)} R_{si}^{(i)} x_{si}^{(i)} / x_{si}^{(i)} - \frac{\sum_{s \in \mathcal{S}(i)} R_{si}^{(i)} x_{si}^{(i)}}{\sum_{s \in \mathcal{S}(i)} x_{si}^{(i)}}.
$$

where $\mathcal{L}(i)$ represents the link set of the $i$th network segment. It can be proved that $T_i$ is also continuous, monotonically increasing, and twice differentiable on the feasible domain.

From the above, we can write the problem of the $i$th segment as

$$
P_i : \max \quad U_{si},
$$

s.t. \quad $r x \leq c^{(i)}$, \quad $x^{(i)} \geq 0$.

According to the properties of $T_i$, by selecting suitable $U_{si}^{(i)}(x_{si}^{(i)})$, for example, $U_{si}^{(i)}(x_{si}^{(i)}) = \log x_{si}^{(i)}$, we can prove that $U_{si}$ is convex on the feasible domain. Thus, $P_i$ is a convex problem [8], that is, it exists only one optimal solution.

B. Reformulation

In this subsection, to solve problem $P_i$ using iterative optimization algorithms, we convert inequality to equality in the constraints. First, we simplify the expression in $P_i$ by vector notations and matrix notations. Then we employ the logarithmic barrier method to reformulate $P_i$.

In the first step, in order to express concisely $P_i$ by vector notations, we represent problem $P_i$ uniformly as

$$
Q1 : \max_{(x)} \quad \bar{U} = (1-w) \cdot \sum_{i=1}^3 \bar{U}_i(x_i) - w \cdot \bar{T}
$$

s.t. \quad $\bar{R} x \leq \bar{c}$, \quad $x \geq 0$,

where $\bar{U}$ denotes the utility of the $i$th segment, $\bar{U}_i$ represents the throughput utility of flow $s$, and $\bar{T}$ denotes the cost of delay. $\bar{R}$ denotes the routing matrix of the $i$th segment. Assume that the set of flows of the network segment is $\bar{S}$, where $\bar{S} \subseteq S$, the number of the set is $\bar{S}$, thus $x$ represents the rate vector, where
\( \mathbf{x} = [x_1, x_2, \ldots, x_5]^T \). Assume that the set of links of the \( i \)th segment is \( \mathcal{L}_i \), where \( \mathcal{L} \subseteq \mathcal{L} \), the number of the set is \( \mathcal{L} \), then \( \mathbf{e} \) represents the link capacity vector, where \( \mathbf{e} = [c_1, c_2, \ldots, c_T]^T \). Consequently, problem \( P_i \) and problem \( Q_1 \) are equivalent.

Next, we reformulate problem \( Q_1 \). We introduce one non-negative slack variable \( y \) whose length is \( \mathcal{L} \) to convert the link capacity constraints to equality constraints. It satisfies \( \mathbf{R} \mathbf{x} + y = \mathbf{e} \), where \( y \) represents the slack capacity of link \( i \). Then, we choose the sequential unconstrained optimization algorithm: logarithmic barrier method, to add the constraints to the target equation. We bring in a decision variable \( z \), which connects flows rates \( \mathbf{x} \) with slack variable \( y \), the form is:

\[
\mathbf{z} = [\mathbf{x}^T, y^T]^T
\]

where the length of \( \mathbf{z} \) is \( \mathbf{S} + \mathbf{L} \).

Thus, the problem \( Q_1 \) can be reformulated as

\[
Q_2 : \min h(\mathbf{z}) = -(1-w) \cdot \sum_{i=1}^{3} \mathbf{U}_i(z) + w \cdot \mathbf{T} - \mu \cdot \sum_{i=1}^{\mathbf{S} \cdot \mathbf{L}} \log z_i
\]

subject to \( F \mathbf{z} = \mathbf{b} \).

where \( \mu \) is a positive coefficient of the barrier function, the vector \( \mathbf{b} = \mathbf{e} \), and \( F \) is a \( \mathbf{L} \cdot (\mathbf{S} + \mathbf{L}) \) matrix defined by \( F = [\mathbf{R} \quad I] \).

Finally, we account for the relationship between problem \( Q_1 \) and \( Q_2 \). We define function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) as \( f(\mathbf{x}) = -\mathbf{U} \), and define \( f^- \) as the negative of optimization for problem \( Q_1 \). For problem \( Q_2 \), we represent the optimal solution as a function of barrier coefficient, namely \( x(\mu) \). It can be shown that when the barrier function coefficient \( \mu \) is close to zero, the optimal solution of problem \( Q_2 \) approaches to the optimal solution of problem \( Q_1 \) [4]. That is, \( f(x(\mu)) \) is close to \( f^- \).

### C. Implementation

Then, we employ quasi-newton algorithms to solve problem \( Q_2 \). Compared with the traditional dual decomposition and sub-gradient algorithm, the quasi-newton algorithms have quadratic convergence rate. In this paper, we mainly employ L-BFGS (Limited-memory BFGS) algorithms. Compared with BFGS, L-BFGS is memory-efficient, which is important for a deep space node with limited resources.

First, to initial quasi-newton algorithms, we need a feasible initial solution which satisfies capacity constraints, where \( \mathbf{z}^0 > 0 \). We use \( \mathbf{z}^k \) to denote the solution vector of the \( k \)th step.

One possible solution is

\[
\mathbf{z}_i^0 = \frac{\min_i \{c_i\}}{S + 1}, i = 1, 2, \ldots, S.
\]

Thus

\[
\mathbf{z}_i^0 = c_i - \sum_{j=1}^{3} \mathbf{R}_{ij} \min_i \{c_i\}, i = 1, 2, \ldots, L,
\]

where \( c_i \) denotes the capacity of link \( i \), and \( R \) represents routing matrix of current segment.

Second, we need to list target equations automatically. According to \( Q_2 \), the target equations \( h(z) \) are composed of \( 2S + L + 1 \) items. Every item can be calculated as follows: we employ suitable function which is continuous, strictly concave, monotonically increasing, and twice continuously differentiable, for example, the logarithmic function to compute throughput utility function \( \sum_{i=1}^{\mathbf{S} \cdot \mathbf{L}} \mathbf{U}_i(z) \), employ formula (5) to compute delay utility function \( \mathbf{T} \), compute directly the rate constraints of flows \( \sum_{i=1}^{\mathbf{S} \cdot \mathbf{L}} \log z_i \), and employ the formula \( \mathbf{z}_i^3 = c_i - \sum_{j=1}^{3} \mathbf{R}_{ij} z_j, i = 1, 2, \ldots, L \) to compute link capacity constraints \( \sum_{i=3}^{\mathbf{S} \cdot \mathbf{L}} \log z_i \).

Next, we can use L-BFGS to solve \( Q_2 \) iteratively. For the optimization problem with \( k \) variables, BFGS needs to save a \( k \times k \) matrix to approximate the inverse of second order hessian matrix. However, L-BFGS only requires to save several vectors with the length of \( k \), thus L-BFGS can save memory efficiently.

### 3.2 Piecewise Rate Allocation Algorithm

In this section, we derive the piecewise rate allocation algorithm (PRA) to solve the original problem \( P \) from the above.

There are four main steps. First, we initialize all the parameters, and define \( D \) as the set of dividing points. The next step is to divide the network into several segments according to the judging value of each node. Then, we solve problem \( Q_2 \) for each segment. And in the process of calculation, we adopt one convenient and compact method which is called two-loop recursion [9] to compute a newton direction. The last step is to obtain the results of the network. The details of PRA are shown as Algorithm 1.
Algorithm 1 PRA

Input: $R$, $e$, link distance $d$
Output: $x$
1) Initialization
   initialize $w$, $\alpha$, $\Delta$, $\mu$, $K$;
2) Segmentation
   for each node $i$ do
      compute $\delta_i$;
      if $\delta_i < \Delta$ then
         put $i$ into the set $D$ of dividing points;
      end if
   end for
3) Solving
   for each segment $j$ do
      compute $R^{(j)}$, $e^{(j)}$;
      initialize feasible solution $z^0$, permissible error $\epsilon$, previous updates size $n$, step $k=0$, position difference $p^0$, gradient difference $q^0$;
      while $\| \nabla h(z^k) \| \geq \epsilon$ do
         use two-loop recursion to compute direction $r^k$;
         search a suitable step size $\beta^k$ that satisfies:
         $$h(z^k + \beta^k r^k) = \min_{\beta \geq 0} h(z^k + \beta r^k) ;$$
         update $z^{k+1} = z^k + \beta^k r^k$;
         if $k > n$ then
            discard vector $p^{k-n}$, $q^{k-n}$ from memory;
         end if
         compute and save:
         $$p^k = z^{k+1} - z^k,$$
         $$q^k = \nabla h(z^{k+1}) - \nabla h(z^k) ;$$
         $k = k + 1$;
      end while
      obtain the optimal rates $x^{(j)}$ of the $j$th segment;
   end for
4) Compute results
   obtain the rates $x = [x^{(0)}, x^{(1)}, x^{(m)}]$.

Then, we analyze the complexity of PRA. Assume that the number of nodes in the network is $k$, the number of segments is $m+1$, the number of iterations of L-BFGS is $k_1$, the search times of step size $\beta$ are $k_2$, $n$ is the size of previous updates in L-BFGS, $S$ is the number of flows in the network, $L$ is the number of links, $S_j$ represents the number of flows in the $j$th segment, and $L_j$ denotes the number of links in the $j$th segment, the time complexity of PRA can be expressed as
$$k + \sum_{j=1}^{m+1} (S_j + k_1 \cdot (n + k_2 \cdot (2S_j + L_j + S_j \cdot L + S_j + L_j)) .$$

Because $k_1$ and $k_2$ are constants, $n < S$, the time complexity can be rewritten as $O(k + S \cdot L)$. Similarly, the space complexity is also $O(k + S \cdot L)$.

4. Experimental Results

To evaluate the algorithm, we conduct experiments with MATLAB. We compare the network utility and end-to-end delays between piecewise rate allocation policy and DNUM policy. In the DNUM policy, we compute the rates of all the flows in the network only once. When link capacity varies during transmission, DNUM may lead to congestion. Thus, we introduce extra cost of delay in DNUM policy.

4.1 Network Utility

We use a linear network and vary the number of links from 4 to 128. There are four flows, as shown in Figure 3.

```
Figure 3. Deep space network topology
```

Table 1. Distance between the earth and other planets

<table>
<thead>
<tr>
<th>Planet</th>
<th>Minimum distance($10^6$km)</th>
<th>Maximum distance($10^6$km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moon</td>
<td>0.3633</td>
<td>0.4055</td>
</tr>
<tr>
<td>Mercury</td>
<td>101.1</td>
<td>221.9</td>
</tr>
<tr>
<td>Venus</td>
<td>39.6</td>
<td>261.0</td>
</tr>
<tr>
<td>Mars</td>
<td>59.6</td>
<td>401.3</td>
</tr>
</tbody>
</table>

According to the distance information between the earth and other planets in Table 1, we choose randomly the distance of links from the range [1.5-10e3, 2.61-10e8].

In accordance with formula (3), we can calculate piecewise points of different networks as follows:

```
Table 2. Dividing points of network

<table>
<thead>
<tr>
<th>Link number</th>
<th>dividing points</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>null</td>
</tr>
<tr>
<td>8</td>
<td>null</td>
</tr>
<tr>
<td>16</td>
<td>$V_8$</td>
</tr>
<tr>
<td>32</td>
<td>$V_8$, $V_{16}$, $V_{24}$</td>
</tr>
<tr>
<td>64</td>
<td>$V_8$, $V_{16}$, $V_{24}$, $V_{32}$, $V_{48}$</td>
</tr>
<tr>
<td>128</td>
<td>$V_8$, $V_{16}$, $V_{24}$, $V_{32}$, $V_{48}$, $V_{64}$</td>
</tr>
</tbody>
</table>
```

In this experiment, we set $\overline{U_j}(z_k) = \log z_k$. In order to reflect the variability of link capacity, for each network
segment, we choose the link capacity uniformly from the range \([10,30]\).

In Figure 4, it shows that when \(w = 0.5\), the utility contrast between piecewise policy and DNUM policy in different networks. When link number is 4 or 8, there is no segmentation, the results of two policies are the same. In other conditions, the network is divided into several segments. We can see that the piecewise policy obtains higher utility than the DNUM policy. With the growth of link number, the improvement increases, up to 35%.

As shown in Figure 6, the utility function with adding delay \(T_i\) can effectively reduce the end-to-end delays in the network. When the number of links is 64 and 128, the utility function with delay \(T_i\) can reduce over 90% delay. Similar results can be obtained by selecting different weight values. This is important for scenarios that require to deliver data in time.

### 4.3 Large-scale Flows

We also examine the running time between piecewise policy and DNUM policy. Considering large-scale scenario with 200 flows and 203 links, the network topology is

![Network topology with 200 flows](image)

Its routing matrix is

\[
R = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & \ldots & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & \ldots & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & \ldots & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & \ldots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & \ldots & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & \ldots & 0 & 1 
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>Policy</th>
<th>Number of iterations</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piecewise</td>
<td>728</td>
<td>2279</td>
</tr>
<tr>
<td>DNUM</td>
<td>587</td>
<td>13144</td>
</tr>
</tbody>
</table>

Table 3. Comparison between piecewise policy and DNUM policy
Table 4. Complexity of BFGS and L-BFGS

<table>
<thead>
<tr>
<th></th>
<th>Time complexity</th>
<th>Space complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFGS</td>
<td>40000</td>
<td>40000</td>
</tr>
<tr>
<td>L-BFGS</td>
<td>16000</td>
<td>8000</td>
</tr>
</tbody>
</table>

According to formula (3), the network is divided into 11 segments. We compute the sum of convergence times and running time of each segment respectively, as shown in Table 3. For the scenario, the running time of the piecewise policy is approximately 1/6 of DNUM policy although it has more convergence times. Moreover, for each segment, the average convergence times are only $728/11 \approx 66$. Table 4 shows the complexity between BFGS and L-BFGS. In this scenario, the required memory of L-BFGS is just 1/5 of BFGS. Thus, we can conclude that for large-scale scenarios, L-BFGS has more obvious advantages than BFGS in terms of storage.

5. Conclusion

Deep space networks have long-haul and intermittent links which result in variable link capacity and routes. In light of the variability, we propose a piecewise rate allocation model so as to obtain optimal rates. Based on the definition of dividing points, we introduce storage cost between adjacent segments. Then, a delay-aware utility objective inside a segment is proposed to consider the trade-off between end-to-end delays and aggregate throughput. Moreover, we derive a rate allocation algorithm which has quadratic convergence rate. Simulation results show that our algorithm achieves higher utility and lower delays compared to DNUM-based algorithms. Meanwhile, in a large-scale scenario, our algorithm is 5 times faster than traditional ones.

Acknowledgements

This research was financially supported by National Natural Science Foundation of China under Grant No.91438119 and National Defense Science and Technique Innovation foundation of Chinese Academy of Sciences under Grant No.CXJJ-15S128.

References