Quadric Lyapunov Algorithm for Stochastic Networks Optimization with Q-learning Perspective

LI Hu*, ZHANG Xinguang1, BAI Jinliang1, SUN Heng1
1 Beijing Institute of Space Long March Vehicle, Beijing, 100076, P. R. China.
*Corresponding author’s e-mail: bupt.foxlee@gmail.com

Abstract. In this article, we investigate stochastic networks optimization using Quadric Lyapunov Algorithm (QLA) with Q-learning perspective. We proposed firstly a model of stochastic queueing networks with power constraints. QLA is then proposed aiming at minimizing an expression containing Lyapunov drift. Based on the analysed similarity between QLA and Q-learning, we show the possibility and feasibility of Q-learning. Simulation of a simple queue network model is carried out, and results using both QLA and Q-learning are compared.

1. Introduction
Stochastic networks optimization deals with optimizing utility function in stochastic queueing networks. Stochastic Queueing Networks is the fundamental model for wired and radio networks with time-varying channels, mobility, and randomly arriving traffic (e.g. link status is not static but typically stochastic). Scheduling algorithm should meet constraints, e.g. nodes in networks typically has transmission power constraints, which means the power for transmission is not infinity at any time. Scheduling algorithms are designed for each layer or cross-layer. [1][2]

The method of Lyapunov optimization can both stabilize the network and deliver a nearly optimal utility behaviour. [1][3] Quadratic Lyapunov function based Algorithm (QLA), which is also known as Backpressure or Max-Weight, is compared with Q-learning algorithm in this article.[4] The method of Lyapunov optimization has been adopted to radio communication nets[4][5], energy gathering nets[6], operation nets[7], monetary systems[8], etc.

Q-learning is a values-based learning algorithm in reinforcement learning that works by learning an action-value function that gives the expected utility of taking a given action in a given state and following a fixed policy thenceforth. A strength with Q-learning is that it is able to compare the expected utility of the available actions without requiring a model of the environment. [9][10]

For stochastic network optimization problems, regardless of technical differences, both QLA and Q-learning make decisions by observing the current network states without knowing the parameters of the environment. Although these two kind of algorithms are proposed in two different fields, it shows that in queueing network scenarios they are quite closely related. On the basis of the above proposed QLA and Q-learning algorithm, the stochastic network optimization problem is the association between QLA and Q-learning.

The remaining of the article is structured as follows. In Section 2, we state our network model and Lyapunov technique. We present results of QLA in Section 3, including the operation of QLA and behaviour of QLA. In Section 4 Q-learning algorithm is proposed and association of those two
algorithms is analysed. Section 5 provides simulation results in a simple queue network model. We conclude in Section 6.

2. Network Model

We describe a mesh model of a stochastic queueing meshes, which characterizes changing topology and force limitations. Suppositions in the model are typical and reasonable for various meshes. [6][7]

The stochastic queueing meshes considered in this article operates in a time-divided manner. Time period \([t, t + t_0]\) is referred as a timestamp \(t\). In the mesh \(\mathcal{N}\) clients in the mesh construct the class \(\mathcal{N}\). Connections between client \(i\) and \(j\), referred as \((i, j)\), construct the class \(\mathcal{L}_i\) and \(\mathcal{N}_i\) refer \(\{a|a \in \mathcal{N}, (a, n) \in \mathcal{L}\}\) and \(\{b|b \in \mathcal{N}, (n, b) \in \mathcal{L}\}\), severally. \(\mathcal{N}\) sorts of packets in the mesh exist \(\mathcal{N}\) sorts of packets. Packets of sort \(c\) designate for \(c \in \mathcal{N}\). Explain the connection condition class as \(\mathcal{S}\). Refer the connection condition of \((i, j)\) at timestamp \(t\), as \(s_{ij}\), while \(s_{ij}'\)s construct the connection condition matrix \(\mathbf{S}(t)\). It is supposed that \(\mathbf{S}(t)\) is i.i.d. and be a value in a finite class \(\{s_1|t = 1, \ldots, M\}\). Suppose that \(p_{ij}(t)\) is limited with a invariant \(P_{\text{max}}\), i.e.

\[
0 \leq p_{ij}(t) \leq P_{\text{max}}, \forall i, j, t
\]

Intermediate force of each client \(i\) is limited fulfilling the restraint

\[
0 \leq \limsup_{t \to \infty} \frac{1}{t} \sum_{t=0}^{t-1} \sum_{b \in \mathcal{N}_i} p_{nb}(t) \leq p_{\text{av}}^i
\]

where \(p_{\text{av}}^i\) is a bounded invariant. Note \(p_{\text{av}}^i \leq P_{\text{max}}\). Velocity of connection \((i, j)\), referred as \(\mu_{ij}(t)\), is conditional on both connection condition \(\mathbf{S}(t)\) and \(\mathbf{P}(t)\), i.e. \(\mu_{ij}(t) = \mu_{ij}(\mathbf{S}(t), \mathbf{P}(t))\). Refer speed of sort \(c\)'s packets of on connection \((i, j)\) by \(\mu_{ij}^c(t)\), so \(\mu_{ij}(t) = \sum_c \mu_{ij}^c(t) \leq \mu_{ij}(t) \leq \mu_{\text{max}}, \forall i, j, t\).

Refer the log of packet \(c\) in client \(n\) as \(Q_n^c(t)\), which construct the matrix \(\mathbf{Q}(t)\). Due to a limited hardware storage \(\mathbf{Q}(t)\) is limited by a comparatively large number, i.e. \(Q_i^c(t) \leq Q_{\text{max}}\). Refer the exogenic arrival velocity to client \(n\) of packet \(c\) as \(R_n^c(t)\), with the supposition that \(R_n^c(t)\) is bounded by a invariant \(R_{\text{max}}\), i.e.

\[
0 \leq R_n^c(t) \leq R_{\text{max}}, \forall i, j, t
\]

Refer the departure velocity assigned to client \(n\) for packet of sort \(c\) at timestamp \(t\) as \(\sum_{t=0}^{t-1} \mu_{\text{out}}(t) = \sum_{b \in \mathcal{N}_n} \mu_{\text{out}}(t)\). Refer \(\max_{\cdot, 0}\) as \(\int_{t}^{t+1} Q_n^c(t)\) evolves according to the undermentioned equation.

\[
Q_n^c(t+1) = [Q_n^c(t) - \sum_{\text{out}} \mu_{\text{out}}(t)]^+, \sum_{\text{in}} \mu_{\text{in}}(t) + R_n^c(t)
\]

Queue \(i\) is average velocity steady (abbreviated for “steady” hereafter) averages:

\[
\mathcal{X}_t = \{X_t | t = 1, \ldots, n\} \text{ refer the virtual queue log vector process of the mesh. Average velocity stableness (abbreviated for “stableness” hereafter) of } \mathcal{X}_t \text{ assures that the related restraint (2) to be practical.}[3]

\[
\text{X}_t(t+1) = [X_n(t) + \sum_{b \in \mathcal{N}_n} p_{nb}(t) - p_{\text{av}}^i]^{+}
\]

Refer \(\mathbf{Q}(t)\) as \((\mathbf{Q}(t), \mathbf{X}(t))\). Utility expression \(U_{\text{tot}}\) is explained as \(\sum_{\text{i,j}} U_{ij}(\mathbf{T}_{ij})\) with the supposition that each \(U_{ij}(\mathbf{T}_{ij})\) is a rigorously monotonic growing convex expression.

We consider a stochastic optimization question with utility maximization.

\[
\max U_{\text{tot}}(\mathbf{F}) \quad \text{s.t. (1):2); Q_n^c(t) \text{ is steady } \forall n, c
\]

Question (6) is equal to question (7) under the conditions of virtual queue mechanics.

\[
\max U_{\text{tot}}(\mathbf{F}) \quad \text{s.t. (5):3); Q_n^c(t) \text{ is steady } \forall n, c; X_n(t) \text{ is steady } \forall n
\]

A mesh attempter is planned to solve question (7), which operates a mesh with a goal of maximizing \(\limsup_{t \to \infty} \mathbf{F}\), subject to the stableness restraint of data queues and virtual queues.

Explain Lyapunov expression as \(L(t) = \frac{1}{2} \sum_{n,c} [Q_n^c(t)]^2 + \sum_n [X_n(t)]^2\). Explain Lyapunov drift as \(\Delta(t) = \mathbb{E}[L(t+1) - L(t)|\mathbf{Q}(t)]\). Explain \(\Delta_{V}(t)\) as \(\Delta_{V}(t) = V\mathbb{E}[[\sum_{n,c} U_n^c(R_n^c(t))|\mathbf{Q}(t)]]\). For \(\Delta_{V}(t)\) we have the undermentioned lemma.

**Lemma 1.** \(\Delta_{V}(t)\) satisfies:
\[ \Delta V(t) \leq B - \mathbb{E}[\sum_{c \in C} U^c_n(R^c_n(t)) - Q^c_n(t) - \mathbb{E}[\sum_{n,c} Q^c_n(t)]] - \mathbb{E}[\sum_{n} \mu^c_n(t) - \mathbb{E}[\sum_{n} X_n(t)\mathbb{P}^{\text{av}} - \sum_{b \in B}\rho_b(t)]] \]

where \( B \) is a positive invariant given in (12).

**Proof.** Squaring both sides of (4) and using the fact \((|x|^2) \leq x^2\), we have
\[
(Q^c_n(t + 1))^2 \leq (Q^c_n(t))^2 + [\sum_{\text{out}} \mu^c_n(t)]^2 - 2Q^c_n(t)[\sum_{\text{out}} \mu^c_n(t) - \sum_{\text{in}} \mu^c_n(t) - R^c_n(t)]
\]

Using the same procedure with (5), we have
\[
(\mathbb{P}^{\text{av}})^2 \leq (\mathbb{P}^{\text{av}})^2 + [\sum_{b \in B}\rho_b(t)]^2 - 2X_n(t)[\mathbb{P}^{\text{av}} - \sum_{b \in B}\rho_b(t)]
\]

Summing (9) over all \( n, c \) and summing (10) over all \( n \) and divided both sides by 2, we have
\[
L(t + 1) - L(t) \leq \frac{N^2}{2} (d^o \mu_{\text{max}})^2 + \frac{N^2}{2} (d^i \mu_{\text{max}} + R_{\text{max}})^2 - \sum_{n,c} Q^c_n(t)[\sum_{\text{out}} \mu^c_n(t) - \sum_{\text{in}} \mu^c_n(t) - R^c_n(t)]
\]

Explain \( B \) as follows: \( B = \frac{N^2}{2} (d^o \mu_{\text{max}})^2 + \frac{N^2}{2} (d^i \mu_{\text{max}} + R_{\text{max}})^2 + \frac{N}{2} (\rho^{\text{av}})^2 + \frac{N}{2} (d^o \rho_{\text{max}})^2 \)

Proof completes by using (12) in (11) and using the definition of \( \Delta V(t) \).

Rescript (8) we have
\[
\Delta V(t) \leq B - \mathbb{E}[\sum_{n,c} V U^c_n(r^c_n(t)) - Q^c_n(t)R^c_n(t)] - \mathbb{E}[\sum_{n} \mathbb{P}^{\text{av}} - \sum_{b \in B}\rho_b(t)]
\]

For a given invariant \( C \geq 0 \), a \( C \)-additive approximation algorithm is one that, every slot \( t \) and given the current \( Q(t) \), chooses a (possibly randomized) action that yields a conditional expected value on the right-hand-side of the drift expression (13) that is within an invariant \( C \) from the infimum. We have the undermentioned lemma. Proof can be found in [4] and omitted for short.

**Lemma 2.** Suppose question (6) is practical and that \( \mathbb{E}[L(0)] < \infty \), if we use a \( C \)-additive approximation of the algorithm every slot \( t \), then: 1) Utility can achieve optimality asymptotically: \( U_{\text{tot}}(\mathbb{F}) \geq U(\mathbb{F}^*) - \frac{C + B}{V} \) where \( B \) is explained in (12). 2) All data queues and virtual queues are steady, and all required restraints in (6) are satisfied.

### 3. QLA for Stochastic Queuing Networks and Theoretical Performance

Explain weight on connection \((i,j)\) at timestamp \( t \) for packets of sort \( c \) as \( W_{ij}^c = \left[ Q_i^c(t) - Q_j^c(t) - \gamma \right]^+ \), where \( \gamma \) is an invariant explained as \( d^i \mu_{\text{max}} + R_{\text{max}} \). Explain weight on connection \((i,j)\) at timestamp \( t \) as \( W_{ij} = \text{max}_c W_{ij}^c(t) \). Refer \( c^* \) as \( \text{argmax}_c W_{ij}^c(t) \). We present here the Quadric Lyapunov Algorithm(QLA) (Algorithm 1).

**Algorithm 1.** QLA for stochastic meshes optimization

REQUIRE: \( X_t \)

ASSURE: Scheduling Policy \((R^c_n(t), P(t), \mu^c_n(t))\)

1. for each timestamp, the mesh attempter does the undermentioned do
   2. Data Admission. Choose \( R^c_n(t) \) to be the optimal of the undermentioned question:
   \[
   \max V U^c_n(r) - Q^c_n(t) r \text{ s.t. } 0 \leq r \leq R_{\text{max}}
   \]

3. Energy Allotment. When \( S(t) = s_t \), choose \( P(t) \) to maximize
   \[
   \max \sum_{n} \sum_{b \in B} \mu_{nb}(P, s_t) W_{nb}(t) - X_n(t) \sum_{b \in B} \rho_{nb}(t)
   \text{ s.t. } p_{nb} \in P_{s_t}, 0 \leq p_{nb} \leq p_{\text{max}}
   \]

   which is equal to
   \[
   \max \sum_{n} \sum_{b \in B} \mu_{nb}(P, s_t) W_{nb}(t) p_{nb}(t) - X_n(t) p_{nb}(t)
   \text{ s.t. } p_{nb} \in P_{s_t}, 0 \leq p_{nb} \leq p_{\text{max}}
   \]
4. Routing & Scheduling. For connection \((n, b)\), if \(W_{nb}(t) \geq 0\), use all energy to transmit packets of sort \(c^*\), i.e. class \(\mu_{nb}^c(t) = \mu_{nb}(t)\), \(\mu_{nb}^c(t) = 0\) \(\forall c \neq c^*\). If terminus is large enough\(Q_b^c(t) + \mu_{nb}^c(t) \geq Q_{\text{max}}\) then terminus client doesn’t store packets, e.g. \(\mu_{nb}^c(t) = 0\).

5. End for

QLA is planned to covetously minimize the upper bound in (13), maximizing utility expression while fulfilling restraints in (7). Thus QLA is a \(C\)-additive approximation algorithm with \(C = C^\text{QLA}\). For the performance of QLA we have the undermentioned theorem.

**Theorem 1.** The undermentioned conclusions to be practical for QLA.
1) We have: 
\[
0 \leq X_n(t) \leq X_{\text{max}} \forall n, t
\]  
(17) Mesh is steady and (2) to be practical.
2) We have \(U_{\text{tot}}(\overline{F}) \geq U(\overline{F}) \geq B + C\) where \(B\) and \(C\) are invariant.
(18)

**Proof.** From (5) it can be seen that \(0 \leq X_n(t) \forall n, t\). From properties of \(\mu_{ij}(t)\) in Section 2, we have
\[
W_{nb}(t)\mu_{nb}(P, s_i) \leq W_{nb}(t)\mu_{nb}(P', s_i) + \eta W_{nb}(t)p_{nb} - X_n(t)p_{nb}
\]  
(19) Suppose \(\eta W_{nb}(t_0) \leq X_n(t_0)\), then we have \(W_{nb}(t_0)\mu_{nb}(P, s_i) - X_n(t_0)p_{nb} \leq W_{nb}(t_0)\mu_{nb}(P', s_i)\), and \(W_{nb}(t_0)\mu_{nb}(P, s_i) - X_n(t_0)p_{nb} = W_{nb}(t_0)\mu_{nb}(P', s_i)\) if and only if \(p_{nb} = 0\). I.e. \(p_{nb} = 0\) maximizes \(W_{nb}(t_0)\mu_{nb}(P, s_i) - X_n(t_0)p_{nb}\). Therefore, if \(\max_{\mu_{nb}(t_0)} \) we have \(p_{nb} = 0 \forall b \in N^n\).

4. Q-learning for Stochastic Queueing Networks

Q-learning algorithm uses \(\epsilon\)-greedy policy to find a balance bewtixt exploitation and exploration. Q-learning behaves covetously most of the timestamp, but with small probability \(\epsilon(0 < \epsilon < 1)\), instead to select randomly from amongst all the actions with equal probability independently of the action value estimates. \(\gamma(0 < \gamma < 1)\) and \(\alpha(0 < \alpha < 1)\) are discount-velocity parameter and step-size parameter severally, controlling the learning speed. Mesh queue logs at timestamp slot \(t\) can be regarded as system conditions \(S(t)\), and mesh policy \((R_i^c(t), P(t), \mu_{nb}(t))\) can be regarded as actions \(\overline{A}(t)\). \(\overline{Q}(t)(S, \overline{A})\) is the current estimate of \(Q\)-table (value of taking action \(\overline{A}\) in condition \(S\) under the optimal policy). Refer the \(\overline{R}(t) = VU_{\text{tot}}(t)\) as the reward expression. We present here the Q-learning Algorithm (Algorithm 2).

**Algorithm 2.** Q-learning Algorithm for stochastic meshes optimization

**REQUER:**\(x_t\)

**ASSURE:** Scheduling Policy \(\overline{A}(t)\).

1. for each timestamp, the mesh attempter does the undermentioned do
2. Action Determination with \(\epsilon\)-Greedy Policy.
3. Choose \(\overline{A}(t)\) to be the optimal of \(\overline{Q}(t)(\overline{S}, \overline{A})\) with probability \(1 - \epsilon\):
\[
\overline{A}(t) = \arg\max_{\overline{A}} \overline{Q}(t)(\overline{S}, \overline{A})
\]  
(20) Choose \(\overline{A}(t)\) to be random with probability \(\epsilon\).

3. Table \(\overline{Q}\) Evolvement. Observe \(\overline{R}(t), \overline{S}(t + 1)\) under action \(\overline{A}(t)\), and evolvement \(\overline{Q}(t)\):
\[
\overline{Q}(t + 1)(\overline{S}, \overline{A}) = \overline{Q}(t)(\overline{S}, \overline{A}) + \alpha[\overline{R}(t) + \gamma\max_{\overline{A}} \overline{Q}(t)(\overline{S}(t), \overline{A}) - \overline{Q}(t)(\overline{S}(t), \overline{A}(t))]
\]  
(21)
5. Simulation

In this section, we provide simulation results of the QLA and Q-learning algorithm in a stochastic queueing meshes with 2 data queues. Queue $i (i = 0$ or $1)$ can determine whether admit $R_i(t)$ ($R_i(t) = 0, 1$ or $2$) packets is at timestamp $t$. Only one queue $i$ can transmit $P_{\text{max}}L_i(t)$ packets, where $L_i(t)$ ($i = 0$ or $0.5$) is the i.i.d. connection status of queue $i$ and $P_{\text{max}}$ ($P_{\text{max}} = 2$) is the uppermost transmission energy. We simulate both QLA and Q-learning algorithm for $10^5$ timestamps with each value of $V$ in the class $\{1, 10, 50, 100, 200, 500, 1000\}$ for 100 timestamps. $\gamma$ and $\alpha$ are set to be invariant ($\gamma = 0.1$, $\alpha = 0.5$), while $\epsilon$ decreases as simulation step grows ($\epsilon = 0.1 - \frac{t}{10^3}$).

Figure 1. Average Data Queue 1 Backlog & Average Data Queue 2 Backlog vs. $V$.

Fig. 1 shows that a linear association betwixt log and $V$ for both QLA and Q-learning algorithm. In addition, intermediate log of Q-learning algorithm is less than that of QLA. Fig. 2 presents intermediate utility of QLA and Q-learning algorithm and a linear association betwixt utility and $V$. From Fig. 1 and 2, it can be inferred that queue log is less but utility is also smaller in Q-learning algorithm than QLA, which averages Q-learning is suboptimal. It is possible that Q-learning algorithm is able to improve performance, e.g. by adjusting the parameters or use more sophisticated reward expression in Q-learning. In Fig. 3 sample paths of $Q_1(t)$ and $Q_2(t)$ when $V = 1000$ are shown. From Fig. 1 and 3, it can be seen that queue log is steady and log of Q-learning algorithm is less than that of QLA.

Figure 2. Average Utility vs. $V$.

6. Conclusion

With regards to quadric Lyapunov algorithms for stochastic queueing meshes optimization question, we firstly proposed a QLA for this sort of meshes. In the next step we analysed Log and utility expression. We proved that under QLA mesh is steady and the condition in timestamp-intermediate limitation could
be met. Q-learning algorithm for stochastic queueing meshes optimization question is the proposed showing the association between QLA and Q-learning algorithm. Simulation results show algorithm are able to stabilize the mesh while trying to optimize the utility expression. It also shows the possibility of improvement of Q-learning.

Figure 3. Sample Paths of Data Queue 1 & Data Queue 2 When $V = 1000$.

References

[1] L. Georgiadis, M. J. Neely, and L. Tassiulas, Resource allocation and cross-layer control in wireless networks. Now Publishers Inc, 2006.

[2] D. B. Nithya Balasubramanian, Thivyavignesh Ramasamy Gurumurthy, “Receiver based contention management: A cross layer approach to enhance performance of wireless networks,” Journal of King Saud University - Computer and Information Sciences, vol. 32(10), no. 10, pp. 1117–1126, 2020.

[3] M. J. Neely, Stochastic Network Optimization with Application to Communication and Queueing Systems, ser. Synthesis Lectures on Communication Networks. Morgan & Claypool Publishers, 2010.

[4] M. J. Neely, “Energy optimal control for time-varying wireless networks,” IEEE Transactions on Information Theory., vol. 52, no. 7, pp. 2915–2934, 2006.

[5] R. Urgaonkar and M. J. Neely, “Network capacity region and minimum energy function for a delay-tolerant mobile ad hoc network,” IEEE/ACM Transactions on Networking, vol. 19, no. 4, pp. 1137–1150, 2011.

[6] L. Huang and M. J. Neely, “Utility optimal scheduling in energy harvesting networks,” in Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing. ACM, 2011, p. 21.

[7] R. Urgaonkar and M. J. Neely, “Utility optimal scheduling in processing networks,” Performance Evaluation, vol. 68, no. 11, pp. 1002–1021, 2011. [8] M. J. Neely, “Stock market trading via stochastic network optimization,” in 49th IEEE Conference on Decision and Control (CDC). IEEE, 2010, pp. 2777–2784.

[9] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. The MIT Press, 2018.

[10] K. Hofmann, “Reinforcement learning: Past, present, and future perspectives,” in Thirty-third Conference on Neural Information Processing Systems (NeurIPS), December 2019, tutorial.