A DATA-DRIVEN APPROACH TO STOCHASTIC NETWORK OPTIMIZATION

Tianyi Chen†, Aryan Mokhtari†, Xin Wang‡, Alejandro Ribeiro†, and Georgios B. Giannakis⋆

†Departments of ECE and DTC, University of Minnesota
‡Department of Electrical and Systems Engineering, University of Pennsylvania
⋆Department of CSE and Key Lab for Inf. Sci. of EMW (MoE), Fudan University

ABSTRACT

This paper considers the long-term network resource allocation problem subject to queue stability. The dynamic problem is first reformulated as a static stochastic programming. To tackle the resultant static programming, we study its dual problem which contains finite number of variables in oppose to the primal problem that has infinite dimension. A novel online framework is developed by formulating the dual stochastic optimization as empirical risk minimization. We first propose an offline scheme for batch training which linearly converges to the optimal dual argument in expectation. The offline approach is further extended to the online setting which successfully converges to the statistical accuracy of the adaptive training set with high probability. It is both theoretically and numerically established that the novel approach can significantly improve delay and convergence of existing network optimization schemes.

Index Terms— Network resource allocation, data-driven, stochastic optimization, statistical learning, machine learning.

1. INTRODUCTION

The recent efforts in data center (DC) network-wide resource allocation aim to mitigate spatio-temporal uncertainties of energy prices, renewable availabilities, and data demands [2–6]. Interestingly, these works reveal that this new engineering problem can still rely on the elegant dual decomposition framework, and the celebrated stochastic network optimization in wireless networking revives its new popularity. The main limitation of these stochastic schemes is their slow convergence rate and high network delay as a by-product. Several attempts have been taken from optimization perspective by accelerating their convergence [7–10]. Yet, none of them exploit the potential benefit of learning from historical data to mitigate future uncertainty.

Turning attention to learning perspective, the proliferation of machine learning advances motivates a systematic way to uncover “hidden” learning through learning from historical relationships and trends in massive datasets [11]. In this context, the current paper revisits the stochastic network optimization problem from a machine learning vantage point, and develops a novel data-driven online approach, which combines the merits of statistical learning and stochastic approximation (queue-based resource allocation). Compared with [12], we focus on an efficient learning protocol to circumvent the computationally expensive procedure in [12] that involves constructing a histogram and solving large-scale optimization per iteration.

The main contributions of this paper are summarized as follows: c1) Targeting a systematic protocol to integrate statistical learning into stochastic resource allocation tasks, we view the resource allocation problem as a machine learning problem that entails batch training and online adaptation phases; c2) Leveraging the special structure of the batch training problem, we generalized the recently developed stochastic average gradient (SAGA) approach in [13] to our training setup, and efficiently compute empirical Lagrange multipliers with an order-optimal linear convergence rate at low computational cost per iteration; and c3) A novel online data-driven approach (that we term online SAGA) is proposed which operates in learn-and-adapt fashion. It is analytically established that the novel online SAGA achieves an improved cost-delay tradeoff relative to existing works in stochastic network optimization [14].

2. NETWORK RESOURCE ALLOCATION

Consider a discrete time index \( t \in \mathbb{N} \), and a cloud network represented by a directed graph \( \mathcal{G} = (\mathcal{J} + I, \mathcal{E}) \). The set \( \mathcal{J} = \{1, \ldots, J\} \) contains mapping nodes (MNs) collecting user data requests, and the set \( I = \{1, \ldots, I\} \) corresponds to the set of DC nodes which process data requests. Further, the set \( \mathcal{E} \) indicates the edge set which contains \( I \cup J \) links. The first \( I \) links (edges) represent links connecting MNs to DCs, and the remaining are "virtual" outgoing links from each DC (e.g., workloads processed in each DC). Define the scalar \( x_{i,j,t} \) as the amount of workload from MN \( j \) to DC \( i \) at slot \( t \), and the scalar \( x_{i,t} \) as the outgoing workloads from DC \( i \). Further, define \( x_t \in \mathbb{R}^{I+J} \) as the global workload vector which is the concatenation of the values \( x_{i,j,t} \) and \( x_{i,t} \).

To enforce the stability of the system, the undistributed (unserved) workload at each MN and DC node requires to be bounded on long-term time average. To do so, we define \( q_t \in \mathbb{R}^{I+J} \) as the concatenation of the queue lengths for all nodes in the network, and \( A \in \mathbb{R}^{(I+J) \times (I+J+I)} \) as the node-incidence matrix of the graph where \( A_{i,e} = 1 \) if link \( e \) enters node \( i \); \( A_{i,e} = -1 \), if link \( e \) leaves node \( i \); and \( A_{i,e} = 0 \), otherwise. By defining \( c_t \in \mathbb{R}^{I+J} \) as the concatenation of stochastic workload arrival rates of the nodes, we can update the length of queues as \( q_{t+1} = [q_t + Ax_t + c_t]^+ \) where the operation \( [\cdot]^+ \) is defined as \( [\cdot]^+ := \max\{\cdot; 0\} \). Moreover, the queue stability is satisfied when all the entries of the expected long-term time
average \((1/T)\sum_{t=0}^{T-1} \mathbb{E}[q_t]\) are bounded for \(T \to \infty\) as in [14].

With \(\Psi_t(x_t):= \Psi(x_t; \psi_t)\) denoting the aggregate network cost per slot \(t\) for resource allocation \(x_t\) and time-varying parameter \(\psi_t\), our goal is to minimize the long-term average network cost subject to queue stability, namely

\[
\Psi^* := \min_{\{x_t, q_t, \psi_t\}} \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\Psi_t(x_t)] \tag{1a}
\]

subject to

\[
q_{t+1} = [q_t + Ax_t + c_t]^+, \forall t \tag{1b}
\]

\[
\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[q_t] < \infty \tag{1c}
\]

\[
x_t \in \mathcal{X}, \forall t \tag{1d}
\]

where the expectation is taken over all sources of randomness (i.e., \(\psi_t, c_t\), and possible randomness of the control policy); and the constraints (1d) instantiate the instantaneous resource allocation within a time-invariant set \(\mathcal{X} \subset \mathbb{R}^{I+J}\); e.g., link transmission limits, capacity limits in DCs.

Notice that the queue dynamics in (1b) couple the optimization variables over infinite time horizon, so finding the optimal solution typically requires dynamic programming tools, which generally suffer from "curse of dimensionality," and are intractable in online setting. We next circumvent this obstacle by relaxing (1b) to limiting average constraints and employing dual decomposition techniques.

Combining (1b) and (1c), it follows that on the average workload arrival and departure rates at each node satisfy the necessary condition \(\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[Ax_t + c_t] = 0\); i.e., all buffered workloads should be served eventually. Consider first \(s_t := [\psi_t^1, c_t]^{\top}\) as the concatenation of all random parameters at step \(t\). By replacing (1b) and (1c) by this limiting average constraints, we can show that if the random process \(s_t\) is stationary, there exists a stationary control policy \(\chi(\cdot)\) which is a pure function of the current \(s_t\); satisfies (1d); and guarantees that \(\mathbb{E}[\Psi_t(\chi(s_t))] = \Psi^*\) with \(\mathbb{E}[Ax(s_t) + c_t(s_t)] \leq 0\); see e.g., [14]. To this end, the dynamic stochastic problem (1) can be relaxed to a static stochastic problem

\[
\tilde{\Psi}^* := \min_{\chi(\cdot)} \mathbb{E}[\Psi_t(\chi(s_t))] \tag{2a}
\]

subject to

\[
\mathbb{E}[Ax(s_t) + c_t(s_t)] \leq 0 \tag{2b}
\]

\[
\chi(s_t) \in \mathcal{X}, \forall s_t \tag{2c}
\]

where we interchangeably use \(\chi(s_t) = x_t\) to emphasize the dependence of the real-time decision \(x_t\) on the random state \(s_t\). Note that the optimization in (2) is with respect to the stationary policy (or the probability distribution) \(\chi(\cdot)\). Hence, there is an infinite number of variables in the primal domain. Observe though, that there is a finite number of constraints coupling the realizations [cf.(2b)]. Thus, the dual problem contains a finite number of variables hinting that the problem is likely more tractable in the dual space [15, 16].

With \(\lambda \in \mathbb{R}^{I+J}\) denoting the Lagrange multipliers associated with constraints (2b) and \(x := \{x_t, \forall t\}\), the partial Lagrangian function of (2) is given by \(L(x, \lambda) := \mathbb{E}[\tilde{L}_t(x_t, \lambda)]\), where the instantaneous Lagrangian is defined as \(L_t(x_t, \lambda) := \Psi_t(x_t) + \lambda^\top (Ax_t + c_t)\). The dual problem of (2) is given by

\[
\max_{\lambda \geq 0} D(\lambda) := \min_{\lambda \geq 0} \mathbb{E}[D_t(\lambda)], \tag{3}
\]

where \(D_t(\lambda) := \min_{s_t \in \mathcal{X}} L_t(x_t, \lambda)\).

If the optimal Lagrange multipliers \(\lambda^*\) are known, a sufficient condition for the optimal solution of (2) is to minimize the Lagrangian function \(L(x, \lambda^*)\) over the set \(\mathcal{X}\) [17, Proposition 3.3.4]. Clearly, the optimal online solution can be characterized as a function of the optimal Lagrange multipliers and the realization of the random state; i.e., \(x_t^* = \arg \min_{s_t \in \mathcal{X}} L_t(x_t, \lambda^*)\), which fits the stationary policy we are looking for (2). However, to implement the aforementioned allocation, the optimal multipliers \(\lambda^*\) must be known. Toward this objective, a novel approach for both offline and online learning is proposed next.

### 3. DATA-DRIVEN NETWORK SAGA

We proceed by stating the required assumptions. These assumptions are actually mild and can typically be satisfied in online network resource allocation problems [7, 12, 18, 19].

**Assumption 1** The random state \(s_t\) has a bounded support, and is independent and identically distributed over time \(t\). Further, there exists a stationary policy \(\chi(\cdot)\) satisfying \(\chi(s_t) \in \mathcal{X}\), for all \(s_t\), and \(\mathbb{E}[Ax(s_t) + c_t(s_t)] \leq -\zeta\), where \(\zeta > 0\).

**Assumption 2** The network cost \(\Psi_t(x_t)\) in (1a) is \(\sigma\)-strongly convex, and its gradient is Lipschitz continuous. The dual function \(D(\lambda)\) in (3) is \(\epsilon\)-strongly concave, and has \(L\)-Lipschitz continuous gradients, with the condition number \(\kappa := L/\epsilon\).

To leverage big historical data, we first postulate a training setting for the considered stochastic optimization problem in (3). Specifically, we approximate (3) by sample averaging on a given training set. Considering the training set \(\mathcal{S} := \{s_n, 1 \leq n \leq N\}\) with \(N\) historical state samples, we can instead solve the following empirical risk maximization problem

\[
\max_{\lambda \geq 0} \hat{D}_N(\lambda) := \max_{\lambda \geq 0} \frac{1}{N} \sum_{n=1}^{N} \hat{D}_n(\lambda), \tag{4}
\]

where \(\hat{D}_n(\lambda) := \min_{s_t \in \mathcal{X}} L_n(x_n, \lambda)\) is the risk with the \(n\)-th sample. The indices \(t\) of the instantaneous Lagrangian and dual functions in (3) are changed to \(n\) to emphasize the dependence on the historical sample \(s_n\).

The objective function in (4) is decomposable as a sum of finite concave functions, and, therefore, we can modify the recently developed SAGA method in [13] to fit our dual learning setup, and efficiently compute empirical Lagrange multipliers. Per iteration \(k\), the offline SAGA first evaluates at the current iterate \(\lambda_k\), one gradient sample \(\nabla D_{\tau}(\lambda_k)\) with sample index \(\tau \in \{1, \ldots, N\}\) selected uniformly at random. Thus, the computational complexity of SAGA is same as that of the stochastic gradient ascent method (SGD) for (4), and markedly less than the batch gradient ascent [13]. However, unlike SGD, SAGA stores a collection of the most recent gradients \(\nabla D_{\tau}(\pi_\tau^k)\) for all samples \(n\), where \(\pi_\tau^k\) represents the most recent iterate for which \(\nabla D_{\tau}(\pi_\tau^k)\) is naturally revealed.

---

1. Generally, the solution to (2) satisfies \(\tilde{\Psi}^* \leq \Psi^*,\) but if one can find an optimality bound w.r.t. \(\Psi^*\) and guarantee the constraints (1b)-(1c) are feasible, the optimality bound w.r.t. \(\Psi^*\) is naturally revealed.

2. The strong concavity is normally assumed in stochastic network optimization [7, 18]. In the worst case, it can be satisfied by subtracting an \(\ell_2\) regularizer \((\epsilon/2)||\lambda||^2\) in (3), for which the sub-optimality is \(O(\epsilon)\) [20, Lemma 3.2].
\( n \)-th gradient sample \( \nabla \hat{D}_n \) is evaluated. The SAGA update for (4) at step \( k \) is given by
\[
\lambda_{k+1} = \left[ \lambda_k + \eta \left( \nabla \hat{D}_r(\lambda_k) - \nabla \hat{D}_n(\pi_k^\tau) + \sum_{n=1}^N \nabla \hat{D}_n(\pi_k^\tau) / N \right) \right]^+
\]
where \( \eta \) denotes the learning rate; the term \( \sum_{n=1}^N \nabla \hat{D}_n(\pi_k^\tau) / N \) is the average of stored gradients in the table; and \( \nabla \hat{D}_r(\lambda_k) \) is the new gradient of the chosen function component with its old version denoted by \( \nabla \hat{D}_r(\pi_k^\tau) \) at iteration \( k \).

For a better illustration of SAGA, consider \( \nabla \hat{D}_r(\lambda_k) \) and \( \nabla \hat{D}_r(\pi_k^\tau) \) as two random variables \( X \) and \( Y \), with the source of randomness \( \tau \). Per iteration \( k \), the main task of SAGA is to efficiently approximate the full gradient of \( \hat{D}_S(\lambda) \) at \( \lambda = \lambda_k \), which is the mean of \( X \); i.e., \( \mathbb{E}[X] = \sum_{n=1}^N \nabla \hat{D}_n(\lambda_k) / N \). For SGD, it simply implements an unbiased estimate \( \nabla \hat{D}_r(\lambda_k) \), with its variance denoted by \( \text{Var}(X) \geq 0 \). In contrast, by leveraging previous gradient evaluations \( \nabla \hat{D}_n(\pi_k^\tau) \), SAGA is able to construct an estimator via [cf. (5)]
\[
\nabla \hat{D}_r(\lambda_k) - \nabla \hat{D}_r(\pi_k^\tau) + \sum_{n=1}^N \nabla \hat{D}_n(\pi_k^\tau) / N
\]
for which the variance will be \( \text{Var}(X) + \text{Var}(Y) - \text{2Cov}(X,Y) \) [21]. Like SGD, SAGA iteration (5) also applies an unbiased gradient estimate; i.e., \( \mathbb{E}[X] = \mathbb{E}[Y] + \mathbb{E}[Y] = \mathbb{E}[X] \). In contrast to SGD, as random variables \( X \) and \( Y \) are gradients for the same function component thus highly correlated (i.e., a large \( \text{Cov}(X,Y) \)), they contribute to a much smaller estimation variance than \( \text{Var}(X) \). Leveraging this virtue, SAGA is able to reduce the iterate variance and get a linear convergence rate.

Under (as2), the convergence rate of the offline SAGA is established in the next theorem.

**Theorem 1.** Consider \( \lambda_k^\tau \) as the optimal solution for the batch learning problem with training set \( S \) and size \( N \). If we choose the learning rate \( \eta = 1/(3L) \), and initialize \( \{ \pi_0^\tau = \lambda_0^\tau, \forall n \} \), the SAGA iteration (5) achieves the linear convergence rate, namely
\[
\mathbb{E}\|\lambda_k - \lambda_k^\tau\|^2 \leq (1 - \rho)^k \left[ \|\lambda_0 - \lambda_k^\tau\|^2 + \frac{2N}{3L} C_S \right]
\]
where the initial error \( C_S = \hat{D}_S(\lambda_0) - \hat{D}_S(\lambda_k^\tau) - \langle \nabla \hat{D}_S(\lambda_k^\tau) \rangle \), \( \lambda_0^\tau - \lambda_k^\tau \); the constant \( \rho = \min \{1/(4N), 1/(3\kappa)\} \) with condition number \( \kappa \), and the expectation is taken over all choices of index \( \tau \) up to iteration \( k \).

Offline SAGA iteration (5) exhibits a competitive performance in terms of convergence rate and computational complexity. However, different from most machine learning tasks, directly implementing empirical Lagrange multipliers from batch to online setting is not applicable, since the online algorithm will lose capability of tracking queue lengths once the system dynamics change. To tackle this limitation, a novel resource allocation approach (termed online SAGA) is proposed next which operates in a learning-while-adapting fashion.

The main idea of the online SAGA in the learning mode is to incrementally increase the training set by including streaming data observed from online resource allocation. However, obtaining an accurate solution for empirical dual problem (4) per new training set (at each slot) is computationally expensive, and online SAGA nicely overcomes this issue in a way that the optimization error of the empirical multiplier \( \lambda_t \) per slot \( t \) approaches the statistical accuracy\(^3\) for the current training set [22]. Hence, online SAGA only needs few extra gradient evaluations per new datum to maintain sufficiently low optimization error.

The online SAGA summarized in Algorithm 1 consists of two complementary stages: offline training and online learning-and-adapting. In the offline stage, Algorithm 1 runs \( K N_{\text{off}} \) SAGA iterations (5) on a training set with \( N_{\text{off}} \) historical samples, where \( K \) is the average number of iterations per datum. With the output of training as a hot start, the online stage keeps learning from dynamic dataset \( S_t \) (steps 8-10) while operating online resource allocation (steps 5-7). Distinct with standard queue-length based resource allocation and machine learning tasks, neither using the queue length \( q_t \) nor the empirical multiplier \( \lambda_t \), SAGA constructs an effective dual variable \( \gamma_t \) for resource allocation, namely
\[
\min_{x_t \in X} L_t(x_t, \gamma_t) \text{, with } \gamma_t = \lambda_t + \mu q_t - \theta, \forall t.
\]
Notice that \( \gamma_t \) is a linear combination of the empirical dual variable \( \lambda_t \) and the instantaneous queue length \( q_t \), where the control variable \( \mu \) tunes the weights of two factors, and the constant \( \theta \) controls the steady-state behavior of \( \gamma_t \), as will be specified in Theorem 3. Unlike standard stochastic dual subgradient (SDGD) or Lyapunov optimization [3, 14], the online SAGA is able to leverage an online learning mode by running offline SAGA in a dynamic training set including new datum \( s_t \).

**Theorem 2.** For the statistical accuracy \( H_s(N) = d / \sqrt{N} \), with \( d > 0 \), consider the training set \( S_t \) at time \( t \) with the sample size \( N_t \) and the empirical optimum \( \hat{D}_S \). If \( K \geq 1 \) and \( N_{\text{off}} \geq 3\kappa/4 \) with condition number \( \kappa \), then the optimization error of online

```python
Algorithm 1 Online SAGA for learning-while-adapting
1: Offline initialize: \( N_{\text{off}} \) historical samples \( S = \{ s_n \} \), dual variable \( \lambda_0 \), intermediate variables \( \pi_0^\tau, \nabla \hat{D}_n(\pi_0^\tau), \forall n, \) and size \( \eta \).
2: Offline mode: run offline SAGA for \( K \times N_{\text{off}} \) iterations in \( S \).
3: Online initialize: initiate a hot start \( \lambda_0, \pi_0^\tau, \nabla \hat{D}_n(\pi_0^\tau), 1 \leq n \leq N_{\text{off}} \) from the output of the offline SAGA, a dynamic dataset \( S \) including \( N_{\text{off}} \) historical samples, the queue length \( q_0 \), and the control variables \( \mu \geq 0 \) and \( \theta = \sqrt{\eta \log^2(\mu); \ldots; \sqrt{\eta \log^2(\mu)}} \).
4: for \( t = 0, 1, 2, \ldots \) do
5: Online adaptive mode:
6: Compute the effective multiplier, and obtain resource allocation \( x_t \) via (8).
7: Update the queues \( q_t+1 \) via (1b).
8: Online learning mode:
9: Add the new sample \( s_t \) to the set \( S_t \) with sample size \( N_t = N_{t-1} + 1 \), and initialize a gradient component \( \nabla \hat{D}_n(\lambda_0) \).
10: With input \( x_t \), run \( K \) times (5) in the new \( S_t \) to output \( \lambda_{t+1} \).
11: end for
```

\(^3\)Statistical accuracy \( H_s(N) = O(1 / \sqrt{N}) \) for a training set with \( N \) samples is a common measure of learnability in statistical learning theory [11]. The learning error is the summation of optimization error and statistical accuracy.
SAGA satisfies
\[
\mathbb{E}\left[ \mathcal{D}_{S_i} - \mathcal{D}_{S_i}(\lambda_t) \right] \leq 7\mathcal{H}'(N_t) + \xi \left( \frac{6\kappa}{\epsilon N_t} \right)^{1.6}, \forall t
\]  
(9)
where the constant \( \xi \) is defined as initial error of (3) w.r.t. the initial iterate \( \lambda_0 \). And the empirical multipliers converge to the optimal argument \( \lambda^* \) of the expected dual problem (3), namely
\[
\lim_{t \to \infty} \lambda_t = \lambda^*, \text{ w.h.p.}
\]  
(10)

Building upon Theorem 2, we can further establish the online resource allocation performance under online SAGA.

**Theorem 3.** Consider \( \Psi^* \) the offline optimal objective of (1) under any feasible control. If the control variable is chosen as \( \theta = \sqrt{\mu} \log^2(\mu) \) with a proper \( \mu > 0 \), the proposed online SAGA yields a near-optimal solution for (1) w.h.p.
\[
\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\Psi_t(x_t(\gamma_t))] \leq \Psi^* + O(\mu)
\]
where \( x_t(\gamma_t) \) denotes the operations obtained from the instantaneous Lagrangian minimization using \( \gamma_t \). Further, the steady-state queue lengths satisfy w.h.p.
\[
\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|q_t\|] = O\left( \log^2(\mu)/\sqrt{\mu} \right).
\]

Theorem 2 entails that for the online learning mode, when the sample size \( N_t \) is sufficiently large, the optimization accuracy with only \( K = 1 \) learning iteration per slot (per new sample) will approach the statistical accuracy \( \mathcal{H}'(N_t) \) and eventually converge to \( \lambda^* \). Theorem 3 further demonstrates that for the online resource allocation, if the control variables \( \mu \) and \( \theta \) are properly chosen, online SAGA is asymptotically optimal with a steady-state queue length \( O(\log^2(\mu)/\sqrt{\mu}) \). This implies that the proposed SAGA approach is able to achieve an optimal \( [\mu, \log^2(\mu)/\sqrt{\mu}] \) cost-delay tradeoff in the context of stochastic network optimization [14]. Comparing with the standard tradeoff \( [\mu, \frac{1}{\sqrt{\mu}}] \) for SDGD or the renowned backpressure, it entails that by leveraging big data, the data-driven design of SAGA can significantly improve the online performance in terms of delay (and empirical convergence). We finally note that if the control variable \( \theta \) is not set appropriately, the effective dual variable \( \gamma_t \) will hover around a value slightly larger than \( \lambda^* \) in steady state thus incurring additional loss, since \( \lambda_t \) will converge to \( \lambda^* \) and queue lengths \( q_t \) are always nonnegative.

**4. NUMERICAL EXPERIMENTS**

The network in tests has \( I = 20 \) DCs and \( J = 20 \) MNs, where each MN has outgoing links to all DCs. Performance is tested in terms of instantaneous queue lengths (delay) and time-average network cost in (1a), namely [cf. model explanations in [1]]
\[
\Psi_t(x_t) := \sum_{i \in T} \alpha_{i,t} (e_{i,t} x_{i,t}^2 - P_t^{c_{i}}) + \sum_{i \in T} \sum_{j \in \mathcal{T}} c_{i,j}^d x_{i,j,t}^2,
\]
where \( \alpha_{i,t} \) is uniformly distributed within \([10, 30]\); coefficients \( \{e_{i,t}\} \) are considered time-invariant as \( \{e_{i,t}\} = \{1, 2, 1, 3, 1, 4, 1, 5\} \); \( P_t^{c_{i}} \) is generated from a uniform distribution within \([10, 50]\); and \( c_{i,j}^d \) is generated from a uniform distribution within \([0, 25, 4]\).
We consider set \( X \) as simple box constraints, the upper limits for \( \{x_{i,t}, \forall i\} \) are set to \([200, 150, 100, 100]\), those for \( \{x_{i,j,t}, \forall i,j\} \) are generated uniformly at random from \([10, 100]\), and all \( x_t \) are lower bounded by 0. For \( e_t \), entries for MNs are generated according to a uniform distribution within \([10, 150]\), and entries for DCs are set to 0. Finally, the offline samples size is \( N_{off} = 1000 \) with \( K = 2 \), and the parameters are \( \theta = \sqrt{\mu} \log^2(\mu) \) with \( \mu = 0.1 \).

We benchmark the online SAGA by the queue-based resource allocation (SDGD) in [3, 14], and SDGD+ which uses a hot start from the offline SAGA, but with same online operations as SDGD. By leveraging offline/online learning steps, it is observable in Fig. 1 that the proposed SAGA converges faster than two alternatives to the optimal operating phase; i.e., it takes SAGA 1.5 × 10^4 slots, but 6 × 10^4 slots for SDGD and SDGD+. In addition, the online SAGA incurs a much lower delay as shown in Fig. 2. For SAGA, the instantaneous network queue length (averages of all nodes) is only 27% of that under SDGD and 32% of that under SDGD+ which only incorporates batch training step. Intuitively, the effective multiplier \( \gamma_t \) has captured both network statistical information from learning step and instantaneous queue variations from testing step, thus SAGA does not need to build up queues to achieve optimal resource allocation, in contrary to SDGD and SDGD+.

**5. CONCLUSIONS**

The stochastic network optimization was revisited in this paper from a machine learning vantage point. An order-optimal learning approach termed SAGA was proposed for learning from batch massive dataset, and its online version was then developed. It was analytically established that the novel data-driven approach achieves an optimal cost-delay tradeoff \( [\mu, \log^2(\mu)/\sqrt{\mu}] \) by leveraging streaming data at the cost of only one more gradient evaluation per new datum.
6. REFERENCES


