Scalable and Distributed Submodular Maximization with Matroid Constraints

Andrew Clark, Basel Alomair, Linda Bushnell, and Radha Poovendran

***Abstract*—Submodular maximization enables efficient approximation of machine learning, networking, and language processing problems. Typically, these problems have been shown to have matroid constraints, which generalize matching and partition conditions. Developing scalable, distributed submodular optimization algorithms that guarantee the same performance as centralized techniques has been an active area of research. In this paper, we address the problem of developing scalable distributed algorithms for submodular maximization with a matroid constraint. Our key step is to construct an auxiliary function from the submodular objective function, and develop distributed exchange-based algorithms for optimizing the auxiliary function. We first introduce a distributed algorithm for maximizing a submodular function with a matroid constraint. We then develop an algorithm for maximizing time-varying submodular functions under partition matroid constraints, which arises in sensor placement and data caching. We prove that both algorithms provide (1-1/e) optimality bounds, and hence achieve the same guarantees as the best centralized algorithms.**

I. Introduction

Submodularity is a diminishing returns property of set functions, analogous to concavity of continuous functions. A variety of problems in machine learning [1], language processing, social networking [2], image processing, and robotics can be modeled as selecting a subset of nodes to maximize a submodular objective function. While subset selection is known to be NP-hard in general, the submodularity property enables efficient algorithms for optimization under a variety of constraints [3], [4].

Matroid constraints generalize cardinality, matching, and linear constraints, and arise naturally in wireless networking applications including sensor scheduling [5], data caching [6], welfare maximization [7], and camera placement. Centralized algorithms with provable optimality guarantees, based on greedy [3], [8] and local exchange [9] techniques, have been developed for maximizing submodular functions with a matroid constraint. The additional structure of a partition matroid constraint, which occurs in the sensor placement, data caching, and welfare maximization problems, has been shown to enable more efficient centralized algorithms with the same optimality properties [3], [7].

A. Clark is with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, WA 01609 USA. aclark@wpi.edu

B. Alomair is with the National Center for Cybersecurity Technology, King Abdulaziz City for Science and Technology, Riyadh, Saudi Arabia. alomair@kacst.edu.sa

L. Bushnell and R. Poovendran are with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA. lb2@uw.edu, rp3@uw.edu

Centralized algorithms for submodular maximization require an entity with global knowledge of the objective function and matroid constraint. In application domains including networking, sensing, and robotics, however, a set of distributed nodes with only local information must decide whether to join or leave the subset using minimal computation, storage, and communication. These constraints have made development of scalable and distributed submodular maximization algorithms an active area of research.

In [5], a distributed greedy algorithm was proposed for cardinality-constrained submodular maximization. This approach requires global time synchronization among the nodes, and also requires each node to periodically broadcast to the rest of the network. Furthermore, while the greedy algorithm achieves the best optimality bound under cardinality constraints, it does not provide optimal performance under general matroid constraints [8]. A distributed submodular optimization algorithm that relies on local exchanges and does not require time synchronization or broadcast was introduced in [10]. This approach, however, does not provide the same optimality guarantees as centralized submodular maximization algorithms. Currently, developing scalable distributed submodular maximization algorithms that achieve these optimality guarantees is an open problem.

In this paper, we address the problem of developing scalable and distributed algorithms for submodular maximization that achieve the same optimality guarantees as the best possible centralized algorithms. In addressing this problem, we make two contributions.

First, we propose a distributed algorithm for maximizing a monotone submodular function subject to an arbitrary matroid constraint. Our approach leverages the fact that optimizing an auxiliary function, instead of the objective function itself, improves the optimality bound [9]. We develop scalable distributed algorithms for computing the auxiliary function, and propose an exchange-based algorithm for optimizing the auxiliary function. We prove that our approach achieves a $(1-1/e)$ optimality bound in polynomial time. We then show how our approach can be applied to commonly occurring classes of matroids, including uniform, partition, linear, transversal, and graphic matroids. Our algorithms rely on local computation and communication between the network nodes, in order to be feasible in resource-constrained wireless networks.

As our second main contribution, we propose distributed online algorithms for maximizing a time-varying monotone submodular function subject to partition matroid constraints. Under our online approach, each node uses an experts algorithm to learn whether to remain in the set, or leave the set and allow a randomly chosen node to join, resulting in a time-varying subset that changes based on the learned objective function. We prove that this online exchange algorithm returns a time-varying set that achieves a (1-1/e) optimality bound compared to the best fixed set.

We demonstrate our approach in a simulation study on data femtocaching in wireless networks. The data caching problem is to select a set of files to store on a collection of data caches in order to maximize the utility of a set of users. This problem was shown to be a submodular maximization problem with a partition matroid constraint in [6], where a centralized greedy algorithm was proposed. Our distributed algorithm converges to the same objective function value as the centralized approach.

The paper is organized as follows. Section II presents related work. Section III describes the system model and assumptions, and gives background on matroids, submodularity, and experts algorithms. Section IV presents distributed algorithms for verifying independence under common classes of matroids. Section V introduces our offline distributed algorithm for submodular maximization with an arbitrary matroid constraint. Section VI presents algorithms for distributed submodular maximization under partition matroid constraints. Section VII contains our simulation results. Section VIII concludes the paper and discusses directions for future work.

II. Related Work

Submodular optimization has been extensively studied in the offline centralized case [3], [8], [11]. In [8], it was shown that a greedy algorithm achieves a $(1-1/e)$ optimality bound for cardinality-constrained monotone submodular maximization and a $1/2$ optimality bound for matroid-constrained monotone submodular maximization. The optimality bound for matroid-constrained submodular maximization was improved to $(1-1/e)$ in [3], where the authors introduced a centralized continuous greedy algorithm.

A purely combinatorial approach to submodular maximization was proposed in [9]. Subsequent works considered submodular maximization under multiple matroid and knapsack constraints, as well as maximization of non-monotone submodular functions [4]. Our approach is based on exchange-based optimization of an auxiliary function as in [9], but does not require value oracle access to the submodular function, which can only be achieved by a centralized entity.

A centralized online algorithm for submodular maximization under a matroid constraint was presented in [12], under the assumption that the algorithm has full oracle access to the objective functions from all previous time periods, which we do not assume.

Distributed submodular maximization was first considered for specific applications including sensor scheduling [5] and data caching [6]. In [5], a distributed online algorithm for submodular maximization was proposed that requires each node to periodically broadcast to all other nodes in the network, limiting the scalability of the approach. In [10], a distributed online submodular maximization algorithm that relies on local exchanges instead of network-wide broadcasts was introduced. This algorithm, however, guarantees an optimality bound of $1/2$, less than the best achievable guarantee of $(1-1/e)$ in the centralized case.

III. Background and System Model

In this section, we present our system model, and then give background on matroids and experts algorithms.

A. System Model

We consider a set of n nodes indexed in the set $V = \{1,\ldots,n\}$. Each node is assumed to be locally time-synchronized with its neighbors, and to have an internal Poisson clock with unit rate (a Poisson clock is a clock that ticks at a set of times $T_1, T_2, \ldots$, where the T_i's are a Poisson process).

The objective function $f(S)$ is assumed to be monotone and submodular, and normalized so that $f(S) \in [0,1]$. The nodes are assumed to have a distributed algorithm to compute $f(S)$. Each node can join or leave the set S at any time t, resulting in a time-varying set indexed S_t. In Section IV, we describe distributed algorithms for the nodes to check whether the matroid constraint $S \in \mathcal{I}$ holds for different classes of matroid. The amount of local information exchange required by each node to compute $f(S)$ and $\mathcal{I}$ depends on the definition of $f(S)$ and $\mathcal{I}$.

B. Matroids and Submodular Functions

A matroid is defined as follows.

Definition 1: A matroid $\mathcal{M}$ is defined by $\mathcal{M} = (V, \mathcal{I})$, where V is a finite set and $\mathcal{I}$ is a set of subsets of V, satisfying (i) $\emptyset \in \mathcal{I}$, (ii) $B \in \mathcal{I}$ and $A \subseteq B$ implies that $A \in \mathcal{I}$, and (iii) A, $B \in \mathcal{I}$ and $|A| < |B|$ implies that there exists $v \in B \setminus A$ such that $(A \cup \{v\}) \in \mathcal{I}$.

If a set A satisfies $A \in \mathcal{I}$, then A is said to be *independent*. We use the notation $A \in \mathcal{I}$ and $A \in \mathcal{M}$ interchangeably to denote independence in matroid $\mathcal{M} = (V, I)$. A maximal independent set is a *basis*. All bases of a matroid will have the same cardinality [13]; the cardinality of a basis is denoted as the *rank* of the matroid.

Lemma 1 ([13]): Let V be a finite set, and define $P_1, \ldots, P_m$ to be a partition of V. Consider a set $\mathcal{I}$ of subsets of V such that $A \in \mathcal{I}$ iff $|A \cap P_r| \leq d$ for all $r = 1, \ldots, m$ and some integer d. Then $(V, \mathcal{I})$ is a matroid, and is denoted as a *partition matroid.*

Other classes of matroids that can be incorporated as constraints in our algorithm are presented in Section IV. The following lemma is needed for our optimality proof, and will be generalized in Section VI.

Lemma 2 ([13]): Let $\mathcal{M} = (V, \mathcal{I})$ be a matroid, and let B_1 and B_2 be bases of $\mathcal{M}$. Then there exists a bijection $\pi : B_1 \rightarrow B_2$ such that, for each $b \in B_1$, $(B_1 - b + \pi(b)) \in \mathcal{I}$, and $\pi(b) = b$ for all $b \in B_1 \cap B_2$.

For any finite set V, a set function $f : 2^V \rightarrow \mathbb{R}$ is *monotone* if for any A and B with $A \subseteq B$, $f(A) \leq f(B)$. The set function $f(S)$ is submodular if for any A and B with $A \subseteq B$, and any $v \notin B$,

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B).$$

C. Background on Experts Algorithms

The goal of an experts algorithm is to select a sequence of actions $s^{(1)}, \ldots, s^{(T)}$ from a fixed set of actions $\mathcal{S} = \{s_1, \ldots, s_L\}$. After choosing each action $s^{(j)}$, the algorithm receives a benefit $u^{(j)}(s^{(j)})$; the goal is to select actions that maximize the total benefit $\sum_{j=1}^{T} u^{(j)}(s^{(j)})$. The experts algorithms will be used in Section VI for online submodular maximization in the case where the objective function changes over time.

An important class of experts algorithms maintain time-varying weights for each action, which are updated at each time step. The exponential weighted updated algorithm, presented as Algorithm 1 below, has been shown to provide both theoretical guarantees and empirical performance [14].

Algorithm 1 Experts algorithm with exponential update.

```
1: procedure ExpWeightedUpdate($\mathcal{A}$)
2:     Input: Set of actions $\mathcal{S}$, with $L = |\mathcal{S}|$
3:     //$w_i$ is weight of node $i$
4:     Initialization: $w_i \leftarrow 1$, $p_i \leftarrow \frac{1}{L}$ for all $i \in \mathcal{S}$
5:         Choose parameters $\alpha$, $\zeta \in [0, 1]$
6:     for Each round $j$ do
7:         Select action $s^{(j)} \in \mathcal{S}$ from distribution $\mathbf{p}$, receive benefit $u^{(j)}(s^{(j)})$
8:         for $i = 1, \ldots, L$ do
9:             if $s^{(j)} = s_i$ then
10:                $\ell_i' \leftarrow (u^{(j)}(s^{(j)}) + \alpha)/p_i$
11:            else
12:                $\ell_i' \leftarrow \alpha/p_i$
13:            end if
14:            $w_i \leftarrow w_i \exp(\zeta \ell_i')$
15:        end for
16:        $\mathbf{p} \leftarrow (\mathbf{1}^T \mathbf{w})^{-1}\mathbf{w}$
17:    end for
18: end procedure
```

The following theorem describes the worst-case optimality guarantees of Algorithm 1.

Theorem 1: The sequence of actions $s^{(1)}, \ldots, s^{(T)}$ chosen by Algorithm 1 satisfy

$$\mathbf{E}\left(\sum_{j=1}^{T} u^{(j)}(s^{(j)})\right) + O\left(\sqrt{(L \ln L)T}\right) \geq \max_{s_i \in \mathcal{S}} \left\{ \mathbf{E}\left(\sum_{j=1}^{T} u^{(j)}(s_i)\right)\right\}. \quad (1)$$

Theorem 1 implies that the EWU algorithm provides a better overall utility than selecting any fixed action at each time step.

IV. Testing Matroid Independence

Our distributed submodular maximization algorithm requires a distributed approach to testing independence in a matroid. In the worst case, determining whether $S \in \mathcal{M}$ for a set S and matroid $\mathcal{M}$ would require each node to know the value of S and have an independence oracle for the matroid, which in turn would require each node to broadcast to the rest of the network when joining or leaving S. For many common matroids, however, more efficient distributed algorithms are possible. Efficient distributed algorithms for common classes of matroids are described as follows.

A. Uniform and Partition Matroids

For uniform matroids, it suffices to show that $|S| = k$, which can be verified by any algorithm that computes $|S|$. Furthermore, the exchange-based algorithm of Section V-C preserves the cardinality at each iteration. For a partition matroid, if each node v knows which partition A_i satisfies $v \in A_i$, then the condition $S - u + v \in \mathcal{M}$ is satisfied iff u and v belong to the same partition and can be verified by u and v alone.

B. Linear and Transversal Matroids

In a more general linear matroid, each node v has a corresponding vector r_v. The set S is independent if the matrix with columns $\{r_v : v \in S\}$ has full rank. In the exchange-based algorithm, by inductive hypothesis S is linearly independent, and hence $S - u$ is linearly independent. It therefore suffices to show that v is not in the span of $S - u$.

One approach to checking whether v is in the span of $S - u$ is by solving the least-squares problem $\min\{||Ax - r_v||_2\}$, where A is the matrix with columns indexed in $S - u$. Distributed least-squares algorithms are readily available in the literature [15]. If the minimum value is sufficiently large, then the vectors are evaluated to be linearly independent.

An important sub-class of linear matroids are *transversal matroids*. A transversal matroid is defined by a set system $A_1, \ldots, A_m$, in which each node v belongs to one or more of the A_i's. A set of nodes S is independent if there is a bipartite matching between S and the sets represented by S. A linear representation can be constructed by each node generating a random vector in $\mathbb{R}^m$, with a nonzero i-th entry if $v \in A_i$ and a zero entry otherwise. A set of nodes S is independent if the corresponding set of columns in the matrix is independent.

C. Graphic Matroids

A matroid is graphic if it can be represented by the collection of forests of a graph, i.e., the set of subsets of edges that do not contain a cycle. Independence in a graphic matroid can be verified by each node determining if it belongs to any cycle, which can be performed in polynomial-time. In particular, if S is a basis, then $S - \{u\} + v$ is independent if and only if u and v belong to the same connected component.

V. Offline Maximization with Matroid Constraint

In this section, we give a distributed algorithm for selecting a set S to solve the problem $\max\{f(S) : S \in \mathcal{M}\}$, where $\mathcal{M}$ is a matroid and $f(S)$ is a monotone submodular function. We let k denote the rank of $\mathcal{M}$. We first define an auxiliary objective function that will be used in the optimization algorithm. We then provide a distributed algorithm for computing the auxiliary function followed by our proposed submodular maximization algorithm. Finally, we analyze the runtime and worst-case optimality bound of our approach.

A. Definition of Auxiliary Function

The auxiliary function $g(S)$ that we employ is the potential function of [9]. The main idea of the auxiliary function is to choose a probability p from a given distribution, and then define a set A that samples each element of S independently with probability p. Intuitively, a set S will have a high value of $g(S)$ if it is "robust", i.e., if it has a high value of $f(\cdot)$ even when some elements of S are removed.

Formally, $g(S)$ is defined as follows. Let $p \in [0,1]$ be chosen with probability density function $\nu(x) = \frac{e^x}{e-1}$. The function $g(S)$ is equal to the expected value of $f(A)$, where A is generated by sampling each element from S with probability p. To arrive at a closed-form expression for $g(S)$, define coefficients $m_{a,b}$ by

$$m_{a,b} = \mathbb{E}(p^b(1-p)^{a-b}) = \int_0^1 \frac{e^p}{e-1}p^b(1-p)^{a-b}\ dp.$$

The function $g(S)$ is then equal to

$$g(S) = \sum_{A \subseteq S} m_{|S|-1,|A|-1} f(A). \tag{2}$$

B. Computation of Auxiliary Function

A centralized algorithm for computing $g(S)$ was proposed in [9]. A distributed algorithm based directly on [9], however, would require all nodes to agree on a value of p at each iteration. While agreement on p could be provided through broadcast or gossip algorithms, this would increase the communication overhead of the algorithm. We present an alternative, distributed approach to computing $g(S)$. The approach requires each node to store the coefficients $\{m_{k,b} : b = 0, \ldots, k\}$, where k is the rank of the matroid. Since changes to the set S occur on a longer timescale than computation of $g(S)$ (through selection of the parameter γ in Section V-C), in what follows we make the simplifying assumption that S is constant during estimation of $g(S)$.

The distributed algorithm maintains a time-varying set $A \subseteq S$. At each tick of its Poisson clock, node $v \in S$ checks whether $v \in A$. If $v \in A$, then v leaves the set A with probability $1/2$, and remains in A otherwise. If $v \notin A$, then v joins A with probability $1/2$, and remains in $S \setminus A$ otherwise.

In order to analyze the time required to estimate $g(S)$, we develop a discrete-time Markov chain model of the set A, in which the discrete time steps occur each time a node's Poisson clock ticks. The states of the Markov chain, denoted $X[m]$, are in the set $\{0,1\}^{|S|}$. Letting v denote the index of the node whose clock ticks at time m, the v-th bit of $X[m]$ is flipped with probability $1/2$ and remains the same with probability $1/2$. Note that $|S| \leq k$, where k is the rank of the matroid, since the set S is assumed to be independent. The following lemma describes the characteristics of this Markov chain.

Lemma 3: The Markov chain $X[m]$ defined above has a stationary distribution in which all states in $\{0,1\}^{|S|}$ have equal weight. This distribution is reached within $O(k \log k)$ ticks. The probability that $X[m]$ and $X[m+M]$ are statistically independent, denoted ρ_M, is bounded by $\rho_M \geq 1 - k^{-\frac{M}{k \log k}}$.

Proof: Transitions can occur between states x and y with $x \neq y$ when the Hamming distance between x and y is equal to 1. Suppose $x_i = 1$ and $y_i = 0$. Then a transition from x to y occurs if i's Poisson clock ticks and i leaves the set A, while a transition from y to x occurs if i's Poisson clock ticks and i joins the set A. Since both events have probability $\frac{1}{2k}$, the detailed balance equations hold and the stationary distribution assigns equal weight to all states.

In order to bound the mixing time of the Markov chain, we use the coupling method, with one walk $X'[m]$ starting from an arbitrary distribution and one walk $X''[m]$ starting from the stationary distribution. The mixing time is bounded above by the time until $X'[m] = X''[m]$. We will have $X'[m] = X''[m]$, in the worst case, after all of the nodes in S have had Poisson clock ticks. This is equivalent to the classical coupon collector's problem with $|S|$ coupons, which has expected time of $|S| \log |S|$. Since $S \in \mathcal{M}$ and k is the rank of $\mathcal{M}$, $|S| \leq k$ and hence the expected time is bounded by $k \log k$.

To prove the second part of the lemma, we observe that for each $i \in S$, $X_i[m]$ and $X_i[m+M]$ will be statistically independent if node i's clock ticks during the interval $\{m, m+1, \ldots, m+M\}$, and will be equal otherwise. Hence, $X[m]$ and $X[m+M]$ will be statistically independent if and only if all k nodes have had Poisson clock ticks during the M total ticks of that interval. Since each node is equally likely to have a clock tick at each time step $m' \in \{m, \ldots, m+M\}$, the probability of this event is equal to the probability that, in the coupon collector problem, all coupons are collected within M samples. This is bounded below by $\left(1 - k^{-\frac{M}{k \log k}}\right)$. ■

The procedure for approximating $g(S)$ is as follows. The set A varies over time as described above. The nodes use the distributed algorithm for computing $f(\cdot)$ to evaluate $f(A)$, and compute $|A|$ using a gossip algorithm, which can run in $O(n \log n)$ time to update $|A|$. After every M clock ticks, each node in V (including nodes not in S) records the current value of $f(A)$ and the value of $|A|$, creating a set of samples $f(A_1), \ldots, f(A_N)$. After N time samples have been recorded, after a total of MN clock ticks, node v computes $\tilde{g}(S) = \frac{1}{N}\sum_{i=1}^{N} m_{k,|A_i|} f(A_i)$.

Theorem 2: Let $\epsilon > 0$, $\delta > 0$. Define N by

$$N = \frac{1}{2}\left(\frac{e}{e-1}\frac{H_k}{\epsilon}\right)\log\frac{1}{\delta},$$

where H_k is the k-th harmonic number, and define M as $M = k \log k \log \frac{N}{\epsilon}$. Then $Pr(|\tilde{g}(S) - g(S)| > \epsilon) < \delta$.

Proof: From [9], N independent samples are needed to ensure that $Pr(|\tilde{g}(S) - g(S)| > \epsilon) < \delta$. Hence, M must be selected to ensure that N independent samples are generated with sufficiently high probability. Let E_i denote the event that the i-th sample is independent of the remaining $(N-1)$ samples. Then we have

$$\begin{aligned} Pr(E_1 \cap \cdots \cap E_N) &= 1 - Pr(E_1^c \cup \cdots \cup E_N^c) \\ &\geq 1 - NPr(E_1^c) \qquad (3) \\ &\geq 1 - Nk^{-\frac{M}{k \log k}} \qquad (4) \end{aligned}$$

where (3) is a union bound and (4) follows from Lemma 3. Substituting the value of M yields the desired result. ■

Theorem 2 implies that the procedure for estimating $g(S)$ can be performed in a distributed manner in $O(M)$ samples.

C. Description of Algorithm

Our proposed algorithm is described as follows. After every γM ticks of its Poisson clock, where M is defined as in Theorem 2 and $\gamma > 1$, each node u takes action depending on whether $u \in S_t$. If $u \in S_t$, then u selects a random node $v \notin S_t$. If $(S_t - u + v) \in \mathcal{I}$, then node u waits for M clock ticks for $\tilde{g}(S_t - u + v)$ to be computed. If $\tilde{g}(S_t - u + v) > (1+\epsilon)\tilde{g}(S_t)$ for a fixed parameter $\epsilon > 0$, then the set S_t is updated to $(S_t - u + v)$. Otherwise, node v leaves S_t and the set is unchanged. The algorithm terminates at node u after a fixed number of iterations L have occurred with $(1+\epsilon)\tilde{g}(S_t) \geq \tilde{g}(S_t - u + v)$ at each iteration.

If node $u \notin S_t$, then node u selects a random node $v \in S_t$ and verifies that $(S_t - v + u) \in \mathcal{I}$. If so, then nodes u and v wait for $\tilde{g}(S_t - v + u)$ to be computed, and determine whether $\tilde{g}(S_t - v + u) > (1+\epsilon)\tilde{g}(S_t)$. The set S_t is updated to $S_t - v + u$ if $\tilde{g}(S_t - v + u) > (1+\epsilon)\tilde{g}(S_t)$ and is unchanged otherwise.

This algorithm is an exchanged-based approach, in which random nodes in S_t are periodically swapped with random nodes in $V \setminus S_t$. The algorithm terminates at each node if a sufficient number of iterations have elapsed with the membership of S_t unchanged, which occurs when the $\tilde{g}$ reaches a local minimum. The optimality bounds and runtime of this approach are analyzed in the following section.

D. Optimality and Runtime Analysis

The worst-case optimality bounds of our approach are characterized in the following theorem.

Theorem 3: Upon termination, the set S_t returned by the algorithm of Section V-C satisfies $f(S_t) \geq (1 - 1/e)f(S^*)$, where $S^* = \arg\max\{f(S) : S \in \mathcal{I}\}$, with probability $1 - \left(\frac{k-1}{k}\right)^L \left(\frac{n-k-1}{n-k}\right)^L$.

Proof: We first show that, with probability $1 - \left(\frac{k-1}{k}\right)^L \left(\frac{n-k-1}{n-k}\right)^L$, the set S_t returned by the algorithm satisfies $g(S_t) > (1+\epsilon)g(S_t - u + v)$ for all $u \in S$ and $v \in V \setminus S$ with $(S_t - u + v) \in \mathcal{I}$. Suppose such nodes u and v exist and are not found by the algorithm. Since the algorithm terminates at nodes u and v after L iterations in which S_t is unchanged, this event occurs if and only if L iterations pass in which u does not select v for a swap and vice versa. The probability that this event occurs is $\left(\frac{k-1}{k}\right)^L \left(\frac{n-k-1}{n-k}\right)^L$.

Based on the above discussion, with probability $1 - \left(\frac{k-1}{k}\right)^L \left(\frac{n-k-1}{n-k}\right)^L$, the set S returned by the algorithm satisfies $g(S) > (1-\epsilon)g(S - u + v)$ for all $u \in S$ and $v \in V \setminus S$. Hence by [9], the set S satisfies $f(S) > (1 - 1/e)\, f(S^*)$. ■

The following theorem characterizes the complexity of our approach.

Theorem 4: Let $g^* = \max\{g(S) : S \in \mathcal{I}\}$ and $g^0 = g(S_0)$. Then the algorithm terminates after $O\left(nL\frac{\log \frac{g^*}{g^0}}{\log(1+\epsilon)}\right)$ iterations, where L is the number of iterations with $(1+\epsilon)\tilde{g}(S_t) \geq \tilde{g}(S_t - u + v)$, for total runtime of $O\left(\gamma MnL\frac{\log \frac{g^*}{g^0}}{\log(1+\epsilon)}\right)$.

Proof: At each iteration, the set S_t changes if and only if the function $\tilde{g}$ improves by a factor of $(1+\epsilon)$. Hence the maximum number of iterations is equal to r satisfying $g(S^*) = (1+\epsilon)^r g(S_0)$, which is equal to $\frac{\log \frac{g^*}{g^0}}{\log(1+\epsilon)}$. In the worst case, each node executes L random exchanges before finding a node that improves $\tilde{g}$, and hence the overall worst-case number of iterations is $nL\frac{\log \frac{g^*}{g^0}}{\log(1+\epsilon)}$. Since each iteration takes γM clock ticks, the total run time is $O\left(\gamma MnL\frac{\log \frac{g^*}{g^0}}{\log(1+\epsilon)}\right)$. ■

We note that the algorithm complexity is a function of $\frac{g^*}{g^0}$. In order to obtain a bound that is independent of g, a distributed algorithm (e.g., the exchange-based approach of [10]) can be used as a first stage to obtain a set S^0 such that g^0 is within a provable $O(1)$ bound of g^* (e.g., a factor of 1/2 in [10]). An analogous approach is taken in the centralized algorithm of [9], where the greedy algorithm is used as a first stage to obtain a $(1 - 1/e)$ gap between g^* and g^0.

VI. Online Maximization with Partition Matroid Constraint

We now study the problem of selecting a time-varying set S_t to solve the problem

$$\begin{array}{ll} \text{maximize} & \int_0^T f_t(S_t)\, dt \\ \text{s.t.} & S_t \in \mathcal{M} \quad \forall t \in [0,T] \end{array}$$

where $\{f_t : t \in [0,T]\}$ is a collection of monotone submodular functions and $\mathcal{M}$ is a partition matroid. We first describe the algorithm, and then analyze its optimality guarantees.

A. Description of Algorithm

Under our approach, each node maintains a separate implementation of an experts algorithm with two actions, denoted s_0 and s_1. After γM ticks of its Poisson clock, where γM is chosen as in Section V-C, each node v first checks whether it is in the set S_t. If $v \in S_t$, then v does not take further action. If $v \notin S_t$, then v sends an exchange request to a randomly selected node $u \in S_t$.

The node u first checks whether $S_t - u + v$ is independent in the matroid. If $S_t - u + v$ is independent, then node u queries its experts algorithm implementation. If the output is s_0, then no action is taken and S_t is unchanged. If the output is s_1, then S_t is updated to $S_t - u + v$. After the decision is made, node u updates the weight according to Algorithm 1 using the benefit $\int_{T_u}^{T'_u} g_t(S_t)\ dt$, where T_u is the current time and T'_u is the next time at which u receives a request.

Intuitively, the algorithm makes exchanges between nodes not in the set S_t and randomly chosen nodes in the set S_t. If the auxiliary objective function g_t is increased by the exchange, then the node that exited the set S_t will be more likely to leave the set S_t in the future. The independence check enforces the condition that the set is feasible (i.e., obeys the matroid constraint) at each iteration.

Algorithm 2 Procedure followed by each node v for distributed online submodular maximization with partition matroid constraint.

```
1: procedure MatroidOpt
2:     for Each clock tick at time $T_v$ do
3:         if $v \notin S_t$ then
4:             Send join request to a random node $u \in S_t$
5:         end if
6:     end for
7:     if $v \in S_t$ and receive join request from $u \notin S_t$ at time $T_v$ then
8:         if $(S_t - v + u) \in \mathcal{M}$ then
9:             Query experts algorithm
10:            if Experts algorithm returns $s_1$ then
11:                $S_t \leftarrow (S_t - v + u)$
12:            end if
13:            Feed back $\int_{T_v}^{T'_v} g_t(S_t)\ dt$ to experts
14:        end if
15:    end if
16: end procedure
```

B. Optimality Analysis

We now analyze the optimality guarantees provided by our approach. As a preliminary step, we give a version of Lemma 2 for a time-varying basis of a partition matroid.

Lemma 4: Let $\mathcal{M}$ be a partition matroid, and let $\{S_t : t \in [0, T]\}$ be a collection of bases of $\mathcal{M}$. Suppose that there exist a set of times $T_1, \ldots, T_l$ such that $S_{T_m} = S_{T_m+} - u + v$ for some u and v, and $S_t = S_{t'}$ for $t, t' \in [T_{m-1}, T_m]$. Let B be a basis of $\mathcal{M}$. Then there exists a family of bijections $\{\pi_t : S_t \rightarrow B\}$ such that, if $S_{t'} = S_t - u + v$ for some u and v, then $\pi_t(u) = \pi_{t'}(v)$, and $S_t - u + \pi_t(u) \in \mathcal{I}$ for all t and $u \in S_t$.

Proof: Let $\mathcal{M}$ be a partition matroid, and let B be a basis. Define $B_i = B \cap P_i$, where P_i is one of the partitions of the ground set V, so that $|B_i| = r$ for some integer r. Index the elements of B_i as $B_{i,1}, \ldots, B_{i,r}$.

Now, let S_t be another basis, and define $S_{ti} = S_t \cap P_i$. S_{ti} and B_i have the same cardinality, and any two elements of S_{ti} and B_i can be exchanged while preserving independence in $\mathcal{M}$. Hence, let $S_{0,i} = \{s_{0,i,1}, \ldots, s_{0,i,r}\}$ be any arbitrary indexing of the elements in S_0, and define $\pi_0(s_{0,i,j}) = b_{i,j}$.

To construct π_t, suppose that $S_{T_m} = S_{T_m+} - u + v$. Since S_{T_m} and S_{T_m+} are both bases, u and v must belong to the same partition. Hence we define $\pi_{T_m+}(u) = \pi_{T_m}(v)$, creating a partition that satisfies the conditions of the lemma. ■

Now, define set B by

$$B = \arg\max\left\{\frac{1}{T}\int_0^T g_t(C)\ dt : C \in \mathcal{M}\right\}.$$

By Lemma 4, for each t, there is a bijection $\pi_t : S_t \rightarrow B$ such that, for each $a \in S_t$, $(S_t - a + \pi_t(a))$ is independent. From these definitions, we state a local optimality result.

Lemma 5: For each $b \in B$,

$$\frac{1}{T}\int_0^T g_t(S_t)\ dt \geq \frac{1}{T}\int_0^T g_t(S_t - \pi_t^{-1}(b) + b)\ dt - O\left(\frac{1}{\sqrt{T}}\right). \quad (5)$$

Proof: Suppose that all nodes $v \neq b$ follow the algorithm of Section VI-A. For these nodes, both the payoffs and independence algorithm are the same as if all nodes (including b) followed the algorithm of Section VI-A. Instead of following the algorithm, node b always chooses s_0.

After some time T_b with finite expectation, node b will join S_t. Now, suppose that at time $t > T_b$, node $v \neq b$ sends a join request. Node v selects a random index in $\{1, \ldots, k\}$ and sends a message to the node in S_t with this index. If the chosen node is not equal to b, then the remaining nodes follow the distributed algorithm, and hence $S_t = \hat{S}_t + b - \pi_t^{-1}(b)$.

If the join request is sent to b, then node b always chooses s_0, and hence the request of node v is refused and S_t is unchanged. Hence, any node that would have occupied index $\pi_t^{-1}(b)$ is excluded from S_t.

By the preceding discussion, the payoff to node b from choosing the fixed action s_0 is

$$\frac{1}{T}\int_0^T g_t(S_t - \pi_t^{-1}(b) + b)\ dt,$$

while the payoff from following the experts algorithm is $\frac{1}{T}\int_0^T g_t(S_t)\ dt$. Hence Theorem 1 implies that (5) holds. ■

Lemma 5 implies that, if each node follows the algorithm of Section VI-A, then the value of the auxiliary function $g_t(S_t)$ is greater than if any fixed node $b \in B$ is added and the corresponding node $\pi_t^{-1}(b)$ removed. We now leverage this local optimality to prove a global optimality result for our approach.

Lemma 6: Let $\pi_t : S_t \rightarrow B$ be the bijection guaranteed by Lemma 4. Then

$$\frac{e}{e-1}\frac{1}{T}\int_0^T f_t(S_t)\ dt + O\left(\frac{k}{\sqrt{T}}\right) \geq \frac{1}{T}\int_0^T f_t(B)\ dt + \frac{1}{T}\int_0^T \sum_{b \in B}\big(g_t(S_t) - g_t(S_t - \pi_t^{-1}(b) + b)\big). \quad (6)$$

Proof: In [9], it was proved that

$$\frac{e}{e-1}f(S) \geq f(B) + \sum_{b \in B}\left[g(A) - g(A - \pi^{-1}(b) + b)\right]$$

where f is a submodular function, g is the corresponding auxiliary function, and $B = \arg\max\{f(S) : S \in \mathcal{M}\}$. Integrating over $t \in [0, T]$ and normalizing by T yields (6). ■

Combining Lemmas 5 and 6 gives the following.

Theorem 5: Let S_t be the time-varying set chosen when all nodes follow the algorithm of Section VI-A. Then

$$\frac{1}{T}\int_0^T f_t(S_t)\ dt \geq (1 - 1/e)\frac{1}{T}\int_0^T f_t(B)\ dt - O\left(\frac{k}{\sqrt{T}}\right).$$

Proof: By Lemma 6, we have that

$$\frac{e}{e-1}\frac{1}{T}\int_0^T f_t(S_t)\ dt \geq \frac{1}{T}\int_0^T f_t(B)\ dt + \frac{1}{T}\int_0^T \sum_{b \in B}\left[g_t(S_t) - g_t(S_t - \pi_t^{-1}(b) + b\right].$$

Substituting Lemma 5 implies that

$$\frac{1}{T}\int_0^T f_t(S_t)\ dt \geq (1 - 1/e)\frac{1}{T}\int_0^T f_t(B)\ dt - (1 - 1/e)kO\left(\frac{1}{\sqrt{T}}\right).$$

■

C. Trading Off Complexity and Optimality

The complexity of the algorithm can be reduced by using $f_t(S_t)$ to compute the benefit for the experts algorithm, instead of the auxiliary function $g_t(S_t)$, since computing $g_t(S_t)$ requires M evaluations of $f_t(S_t)$. Under this approach, lines 11 and 17 of Algorithm 2 are replaced with $\int_{T_v}^{T_v'} f_t(S_t)\ dt$. The following lemma describes the local optimality.

Lemma 7: For the simplified distributed optimization algorithm, the set S_t that is returned satisfies

$$\frac{1}{T}\int_0^T f_t(S_t)\ dt \geq \frac{1}{T}\int_0^T f_t(S_t - \pi_t^{-1}(b) + b)\ dt - O\left(\frac{1}{\sqrt{T}}\right)$$

for each $b \in B$, where $\pi_t : S_t \rightarrow B$ is the bijection guaranteed by Lemma 4.

The proof is analogous to the proof of Lemma 5 and is omitted. The following lemma then describes the global optimality of the simplified algorithm.

Theorem 6: For the simplified distributed optimization algorithm, the set S_t that is returned satisfies

$$\frac{1}{T}\int_0^T f_t(S_t)\ dt \geq \frac{1}{2T}\int_0^T f_t(B)\ dt - O\left(\frac{k}{\sqrt{T}}\right) \qquad (7)$$

The proof is omitted due to space constraints.

VII. Simulation Study

In this section, we give the results of our simulation study. We first describe the application, and then present the simulation results.

A. Simulation Setup

We consider the distributed data caching application in [6]. A set of files $F = \{f_1, \ldots, f_M\}$ are accessed by a group of users in a wireless network. The data are stored on a set of distributed caches $C = \{c_1, \ldots, c_k\}$; due to storage constraints, however, only h files can be kept in each cache. The set that must be selected is defined by the set of data to be stored on each cache.

In [6], the problem is mapped to submodular maximization with a matroid constraint. The ground set $V = \{\alpha_{ij} : i = 1, \ldots, M, j = 1, \ldots, k\}$, where $\alpha_{ij} \in S$ implies that cache j stores file i. Each cache's storage constraint is mapped to a partition matroid, i.e., $|\{\alpha_{ij} : j = j'\}| \leq h$ for all j'. Let $\mathcal{M} = (V, \mathcal{I})$ denote this matroid. We have that $S_t - \alpha_{ij} + \alpha_{i'j'} \in \mathcal{I}$ iff $j = j'$. Hence the only feasible exchanges consist of removing one file and adding a different file to the same node, and matroid independence can be verified efficiently.

The time for a user to access the file is equal to the number of hops between the user and the nearest cache holding the file. Hence, the objective function is determined by the probability that a user can access a desired file from a cache within the user's local neighborhood. Let C_m denote the set of caches that are within radio range of user m, and let $F_m = \{\alpha_i : \alpha_{ij} \in S \text{ for some } j \in C_m\ \}$. Formally, the objective function $f(S)$ is defined by $f(S) = \sum_{m=1}^{k}\sum_{\alpha_i \in F_m} P_i$ where P_i is the utility of accessing file α_i. This function was shown to be submodular in [6]. The objective function can be computed by each cache storing the set of users that connect to it.

For the simulation study, we considered a network with 20 users, $k = 3$ caches, $W = 10$ files, and the number of files per cache varied from $h = 1$ to $h = 8$. The users and caches were placed uniformly at random, within a deployment area chosen such that each user could connect to on average 2 caches. We compared the centralized greedy algorithm, proposed in [6], with the distributed exchange-based submodular maximization algorithm presented in Section V-C.

B. Simulation Results

We first investigated the rate of convergence, as well as the final objective function value, for the distributed algorithm and compared with the centralized approach (Figure 1(a)) when $k = 3$. We found that after 100 iterations, the distributed algorithm had reached within 0.05 of the centralized approach. The objective function value oscillated within a neighborhood of the centralized value; the oscillation was due to variations in the potential function $g(S)$, which led to elements with similar potential function values being exchanged repeatedly. We observed a similar convergence behavior as the number of files per cache increased. These oscillations could be reduced by increasing the number of random samples used to compute the potential function, but at the cost of additional complexity.

We compared the final objective function achieved by the centralized and distributed algorithms for different cache sizes (Figure 1(b)). For small cache sizes, the distributed approach provides the same objective function value, while there is a gap of less than 5% when as the cache size grows larger. This

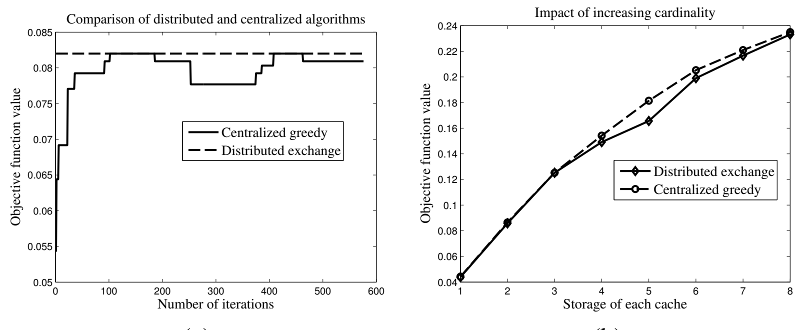

Fig. 1. Simulation study of distributed submodular maximization in a data caching application, in a network with 20 users, $k = 3$ caches, $M = 10$ files, and a varying number of files per cache. The study compares the centralized greedy algorithm with our distributed exchange-based approach. (a) Convergence of distributed algorithm to centralized value. The exchange-based algorithm oscillated in a neighborhood of the centralized value, due to the inherent randomness in computing the potential function $g(S)$. (b) Comparison of distributed and centralized algorithms as the number of files per cache, and hence the cardinality of S, increases. The distributed approach remains close to the utility achieved by the centralized algorithm as the number of files increases.

gap was reduced by reducing the value of ϵ in the algorithm of Section V-C, at the cost of additional iterations.

VIII. Conclusions and Future Work

In this paper, we studied distributed submodular maximization subject to matroid constraints, in which each node decides whether to join a time-varying optimized set in order to maximize a submodular objective function. We made two main contributions toward the problem studied. Our first contribution is an offline distributed algorithm for maximizing a fixed submodular objective function with an arbitrary matroid constraint. Under our proposed algorithm, a node in the set is exchanged with a node not in the set, and the impact on an auxiliary function is computed by the network nodes. The auxiliary function of a set is obtained by computing the objective function at random subsets of the sets, which determines whether the objective function is robust to nodes being removed from the set. We show that this auxiliary function can be computed efficiently, and that the distributed exchange-based algorithm leads to a $(1 - 1/e)$ optimality bound, which is equal to the best achievable optimality bound.

We then considered online submodular maximization, in which the objective function varies over time, under partition matroid constraints. Partition matroids are a widely-studied class of matroids that generalize cardinality constraints [13] and the submodular welfare problem [7], and have direct applications including data cache placement [6]. In our online algorithm, each node in the time-varying set estimates the value of being replaced by a randomly chosen node using an experts algorithm. We proved that the online algorithm satisfies a local optimality condition, namely, that the time-varying set chosen by the algorithm provides a larger objective function value than a set in which any fixed node is included at each time step. Based on the local optimality result, we then showed that the online algorithm achieves a $(1-1/e)$ optimality bound compared to the best possible fixed set, in the limit as the number of time steps goes to infinity.

An important direction of our future research is to generalize the distributed online algorithm from partition matroids to arbitrary matroids. We also plan to research and develop scalable and distributed algorithms for submodular maximization with knapsack or multiple matroid constraints.

References

[1] A. Kulesza and B. Taskar, "Determinantal point processes for machine learning," *arXiv preprint arXiv:1207.6083*, 2012.
[2] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 137–146, 2003.
[3] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrak, "Maximizing a submodular set function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.
[4] J. Lee, V. Mirrokni, V. Nagarajan, and M. Sviridenko, "Maximizing non-monotone submodular functions under matroid or knapsack constraints," *SIAM Journal on Discrete Mathematics*, vol. 23, no. 4, pp. 2053–2078, 2010.
[5] D. Golovin, M. Faulkner, and A. Krause, "Online distributed sensor selection," *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 220–231, 2010.
[6] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," *Proceedings of IEEE Infocom*, pp. 1107–1115, 2012.
[7] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 67–74, 2008.
[8] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions–II," *Polyhedral combinatorics*, pp. 73–87, 1978.
[9] Y. Filmus and J. Ward, "Monotone submodular maximization over a matroid via non-oblivious local search," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 514–542, 2014.
[10] A. Clark, B. Alomair, L. Bushnell, and R. Poovendran, "Distributed online submodular maximization in resource-constrained networks," *12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pp. 397–404, 2014.
[11] H. Narayanan, *Submodular Functions and Electrical Networks.* Elsevier, 1997, vol. 54.
[12] D. Golovin, A. Krause, and M. Streeter, "Online submodular maximization under a matroid constraint with application to learning assignments," *arXiv preprint arXiv:1407.1082*, 2014.
[13] J. Oxley, *Matroid Theory.* Oxford University Press, 1992.
[14] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games.* Cambridge University Press, 2006.
[15] L. Xiao, S. Boyd, and S. Lall, "A space-time diffusion scheme for peer-to-peer least-squares estimation," *Proceedings of the 5th international conference on Information processing in sensor networks*, pp. 168–176, 2006.