Completely Uncoupled Algorithms for Network Utility Maximization

S. Ramakrishnan and Venkatesh Ramaiyan

Abstract—In this paper, we present two completely uncoupled algorithms for utility maximization. In the first part, we present an algorithm that can be applied for general non-concave utilities. We show that this algorithm induces a perturbed (by $e$) Markov chain, whose stochastically stable states are the set of actions that maximize the sum utility. In the second part, we present an approximate sub-gradient algorithm for concave utilities, which is considerably faster and requires lesser memory. We study the performance of the sub-gradient algorithm for decreasing and fixed step sizes. We show that, for decreasing step sizes, the Cesaro averages of the utilities converges to a neighborhood of the optimal sum utility. For constant step size, we show that the time average utility converges to a neighborhood of the optimal sum utility. Our main contribution is the expansion of the achievable rate region, which has not been considered in the previous paper on completely uncoupled algorithms for utility maximization. This expansion aids in allocating a fair share of resources to the nodes, which is important in applications like channel selection, user association, and power control.

Index Terms—Distributed resource allocation, learning in games, utility maximization and fairness.

I. INTRODUCTION

RADIO resource allocation is an important problem in infrastructure, ad-hoc and sensor networks [1]. In particular we need to address the following resource allocation problems, viz., channel selection, user association and power control. Channel selection and power control are essential for the efficient use of radio resources, whereas user association deals with efficient use of deployed Access Points. The solution should cater to the following objectives: (i) Network throughput optimality be ensured (ii) Users get a fair share of the network throughput (iii) Easily implementable. With the advances in 5G wireless systems, it is predicted that there will be a phenomenal increase in the number of access points [2]. Added to this, we have coexisting radio technologies like LTE and Wifi [3]. In such scenarios, centralized solution is unsuitable due to a large overhead. Further, centralized control is also impractical in a heterogeneous set-up. Thus robust and easy to implement distributed solutions are desirable.

In this paper, we provide solutions to the above problems with the stated objectives. We arrive at such a solution using the approach in [4], which the authors call as completely uncoupled learning. In completely uncoupled learning, nodes’ decisions are based only on their past actions and utilities. Marden et al. [5] proposed a completely uncoupled algorithm that maximizes the sum pay-off (which translates to maximizing sum throughput in wireless networks). An important attribute to consider in radio resource allocation is fairness among nodes, i.e., every node should get a fair share of the network throughput. In this paper, we consider the problem of utility maximization, where maximizing some utility functions have a notions of fairness [6]. We propose two completely uncoupled algorithms that maximize the sum utility of the nodes. In our first algorithm, we discretize the rate region, thereby we pose the utility maximization as a combinatorial optimization problem. This algorithm applies to general utilities, not necessarily concave. In our second algorithm, we propose an approximate sub-gradient algorithm for maximizing concave utilities. The main contribution of our work is to provide flexibility in operating at any point in the interior of the rate region.

Our algorithms are general and can be applied to any general network utility maximization not restricted to wireless networks. We present our algorithm for a general network, while bearing the above stated applications in mind.

A. Related Literature

Tassiulas and Ephremides [7] proposed the max weight algorithm. The max weight algorithm can stabilize any arrival rate within the rate region [7]. Proportional fair scheduler was shown to optimize logarithmic utility function in [8]. Neely et al. [9] proposed an algorithm that could stabilize any arrival rate within the rate region and optimize a concave utility for arrival rates exceeding the rate region. The main drawback of the max weight algorithm, used in [7]–[9], is its complexity and centralized nature.

Maximal scheduling algorithms, having low complexity, could support only a fraction of the rate region [10]. Greedy algorithms such as longest queue first scheduling, are optimal only for a class of network topologies [11]. Distributed algorithms based on Gibbs sampling were proposed for IEEE 802.11 WLANs in [12], for channel selection and user association. A proportional fair resource allocation algorithm for channel selection and user association was proposed in [13]. Both [12] and [13] require neighbor information exchange (or knowledge) and are applicable only to some tailored utilities.

In [14], Jiang and Walrand proposed distributed scheduling algorithms for a conflict graph model without collisions. They proved that their algorithms are optimal assuming time scale.
separation. Liu et al. [15] showed that stochastic approximation [16] leads to time scale separation when the update parameters are bounded. Ni et al. [18] proposed distributed scheduling algorithms and showed them to be optimal without time scale separation or bounded parameters assumption. A discrete time version called Q-CSMA was proposed by Jian Ni et al in [18], where collision free schedules are generated by considering a control phase, thereby allowing multiple links to change their state. These results were extended to more practical SINR models in [19] and [20]. However, the pay-off of a node is assumed to depend on the actions of other nodes only through a feasibility constraint. In our work, we allow the pay-off to be a general function of the joint action profile.

Resource allocation problems have been studied as cooperative and non-cooperative games. In repeated prisoners dilemma with selfish players, Pavlov method was used to adapt a player’s strategy in response to other players’ strategies [21].

Marden et al. [22] formulated cooperative control problems as a repeated potential game. They designed objective functions for players, such that the log-linear learning rule converges to a pure Nash equilibrium in a probabilistic sense, where Nash equilibrium action is played for a large fraction of time. The idea of state based potential games was introduced in [23] with the goal of designing local objectives to attain a desired global objective through log-linear learning, where the introduction of state helps in coordination.

A completely uncoupled algorithm to reach efficient Nash equilibrium was proposed by Pradelski and Young [4]. The algorithm was based on the theory of perturbed Markov chains [24], [25]. With similar ideas, Marden et al proposed algorithms to achieve maximum sum pay-off in [5]. These algorithms were adapted to wireless networks in [26]. Borowski et al. [27] proposed distributed algorithm to achieve efficient correlated equilibrium. In our prior work [28], we proposed a distributed algorithm for utility maximization and used perturbed Markov chain ideas to prove optimality. In [29], we extended [28] to state based models.

**B. Contributions**

1) In this paper, we propose two distributed algorithms for utility maximization. To the best of our knowledge, we are the first to propose completely uncoupled utility maximization algorithms that achieve the entire rate region. These algorithms find application in distributed channel selection, user association and power control in a variety of wireless networks.

2) In the first algorithm, which we call General Network Utility Maximization (G-NUM), we allow the utilities to be general functions (not necessarily concave) of the average pay-off. We show that G-NUM is optimal and the sum pay-off maximizing algorithm in [4] is a special case of it.

3) For concave utilities, we propose our second algorithm, Concave Network Utility Maximization (C-NUM). In C-NUM, we present an approximate subgradient algorithm inspired by Gibbs sampling based Utility maximization algorithm in [14]. With C-NUM, we show an important connection between completely uncoupled algorithms based on perturbed Markov chains and Gibbs sampling based utility maximization algorithms such as [14].

4) We also derive upper bounds on the mixing time for the algorithm in [4] and show that the mixing time (upper bound) grows exponentially in the number of nodes.

**C. Outline**

The rest of the paper is organized as follows. In Section II, we discuss the system model. In Section III, we propose our first algorithm on general utility maximization and discuss convergence results. We present the second algorithm for concave utilities in Section IV with convergence results. We discuss numerical results in Section V. In Section VI, we provide a summary with comparisons. The proofs of our results are discussed in detail in the Appendix.

**II. SYSTEM MODEL**

We consider a system of $N$ nodes. The set of nodes is denoted by $\mathcal{N}$. We assume a slotted time model. In time slot $t$, node $i$ chooses to play an action $a_i(t) \in \mathcal{A}_i$. We assume that, for all $i$, $\mathcal{A}_i$ is finite. Let $a(t) = (a_1(t), a_2(t), \ldots, a_N(t)) \in \mathcal{A}$ denote the action profile at time $t$, where $\mathcal{A} = \prod_i \mathcal{A}_i$. In slot $t$, node $i$ gets a pay-off $r_i(t)$. We assume that,

$$r_i(t) = f_i(a(t)),$$

where $f_i$ is a non-negative bounded function from $\mathcal{A} \to \mathbb{R}^+$. Without loss of generality, we assume that $f_i$ is bounded between 0 and 1. We allow $f_i$ to be general, thereby allowing our setup to be applied in a variety of models. Let $p(a)$ denote the fraction of time action profile $a \in \mathcal{A}$ is chosen, where $\sum_a p(a) = 1$. The average pay-off received by node $i$ is given by,

$$s_i(p) = \sum_a p(a) r_i(a).$$

Let $s = (s_1, s_2, \ldots, s_N)$ denote the vector of average pay-offs obtained by the nodes. We say that a pay-off vector $s$ is achievable, if there exists a $p > 0$ satisfying (1). We denote by $\mathcal{S}$, the set of achievable pay-offs. Then,

$$\mathcal{S} = \left\{ s = (s_i(p)) | \sum_a p(a) = 1, p(a) \geq 0 \right\}.$$ 

Let $U_i$ denote the utility of node $i$, where $U_i$ is a function of $s_i$. Without loss of generality, we assume that $U_i$ is bounded between 0 and $u_{\text{max}}$, where $u_{\text{max}} < 1$. The objective here is to maximize the sum utility, $\sum_i U_i(s_i)$, where $s \in \mathcal{S}$. Alternatively, the utility maximization problem is posed as follows,

$$\max \sum_i U_i(\bar{r}_i)$$

s.t. $\bar{r}_i \leq \sum_a p(a) r_i(a)$,

$$\sum_a p(a) = 1, \quad p(a) \geq 0. \quad (2)$$
In this work, we seek a distributed algorithm that solves (2). By distributed, we mean that the algorithm is completely uncoupled, where every node has knowledge only about its previous actions and pay-offs. A node chooses its action purely based on its previous actions and pay-offs. We assume that the network satisfies the following interdependence definition.

Assumption 1: Interdependence: For any proper subset of the nodes $M \subset N$ and any action profile $a = (a_M, a_{-M}) \in A$, where $a_M \in \prod_{i \in M} A_i$ and $a_{-M} \in \prod_{i \in N \setminus M} A_i$, there exists a node $j \in N \setminus M$ and a choice of actions $a_M, a_{-M} \in \prod_{i \in M} A_i$ such that $f_j(a_M, a_{-M}) \neq f_j(a_M, a_{-M})$.

Remark 1: A key assumption for our algorithm to work is interdependence. We study a completely uncoupled setup, where the only feedback to a node on the action profile is its pay-off. Interdependence ensures that changes in actions by any node(s) can be perceived by other nodes in the network as a change in pay-off.

Alternatively, the adaptive CSMA models in [14] and [18] assume a conflict graph network model. Extensions of [14] and [18] to more practical SINR interference model was considered in [19], [20]. In the above works, the service rate of a link depends on the actions of other links only through the notion of feasible actions (transmission modes) i.e., $r_j(a) = f_j(a)$ s.t. $a_i$ is feasible. In contrast, we allow pay-offs to be a function of the joint action profile i.e., $r_i(a) = f_i(a)$, where $a = (a_1, \ldots, a_N)$ is the joint action profile. Such an assumption is preferred in applications such as user association and channel selection (See models in [13] and [30]). In our work, we assume that the network satisfies interdependence, which enables us to work with a more general model.

### III. General Network Utility Maximization Algorithm

In this section, we present a completely uncoupled utility maximization algorithm for general utilities (possibly non-concave) and discuss its convergence results. Algorithm 1, which we call General Network Utility Maximization (G-NUM) algorithm is described below.

The history (and possible state) of any node $i$ at the end of slot $t-1$ is the sequence of actions $(a_i(1), \ldots, a_i(t-1))$ and the pay-offs received $(r_i(1), \ldots, r_i(t-1))$. We require that the nodes maintain an internal “satisfaction” variable, denoted by $q_i(t-1)$ (of node $i$, at time $t-1$), which is a function of the action and the pay-off received in the previous $K$ slots (where $K$ is a fixed positive integer). We let $q_i$ take values from the binary set $\{0,1\}$, where $q_i = 1$ represents a state of “content” with the choice of actions and the pay-off received (in the previous $K$ slots), while $q_i = 0$ represents a state of “discontent” for the node. For every slot, the nodes have to choose an action $a_i$ and update their satisfaction variable $q_i$. Let $q = (q_1, \ldots, q_N)$ denote the vector of satisfaction variables.

Node $i$ chooses action $a_i(t)$, at the beginning of slot $t$ depending on its satisfaction variable $q_i(t-1)$. If node $i$ is content at the beginning of slot $t$, i.e., $q_i(t-1) = 1$, then it repeats an earlier action, here $a_i(t-K)$, with high probability $1 - \epsilon^c$ (where, $c$ is a parameter and $c > N$). With a small probability $\epsilon^c$, any other action is chosen uniformly at random. When node $i$ is discontent, i.e., $q_i(t-1) = 0$, the node selects an action randomly and uniformly from $A_i$.

The satisfaction variable $q_i(t)$ is updated by the end of slot $t$. If the node $i$ was content in slot $t-1$ (i.e., $q_i(t-1) = 1$), then, it continues to remain content if it had repeated an earlier action (i.e., if $a_i(t) = a_i(t-K)$, which happens with high probability) and received the same average pay-off (average over the last $K$ slots), i.e., $\sum_{j=t-K+1}^{t} r_i(j) = \sum_{j=t-K+1}^{t} r_i(j)$ (which would happen when the action profile in the network repeats every $K$ slots). Otherwise, a node becomes content with a very low probability depending on the utility, where the utility is a function of the average of the pay-offs received in the last $K$ slots.

The satisfaction variable aids in synchronizing changes in actions across the network. When all the nodes are content, i.e., $q_i(t) = 1$ for all $i$, the nodes continue to repeat the last $K$ actions (in synchrony) and continue to receive a constant average pay-off based on the sequence of actions. Now if a node decides to change its action, it becomes discontent with a large probability and chooses its action randomly in the subsequent slots. By interdependence, with a large probability, this causes other nodes in the network to become discontent and sets a ripple effect causing all nodes in the network to become discontent. Finally, the nodes become content again, where a sequence of action profiles is preferred depending on the average pay-off corresponding to the $K$-sequence and nodes’ utilities.

In the remainder of this section, we will prove that the above algorithm chooses an action sequence that optimizes the formulation in $(2)$ as $\epsilon \to 0$ and as $K \to \infty$.

---

**Algorithm 1 General Network Utility Maximization Algorithm (G-NUM)**

**Initialize:**
- Fix $c > N$, $\epsilon > 0$ and $K \geq 1$.
- For all $i$, set $q_i(0) = 0$.

**Action update at time $t$:**
- If $(q_i(t-1) = 1)$, then
  - $a_i(t) = \begin{cases} a_i(t-K) & \text{w.p. } 1 - \epsilon^c \\ a_i \in A_i & \text{w.p. } \epsilon^c \end{cases}$ if $a_i \neq a_i(t-K)$
- Else
  - $a_i(t) = a_i \text{ w.p. } \frac{1}{|A_i|}$ where $a_i \in A_i$

**Update for $q_i$ at time $t$:**
- If $(q_i(t-1) = 1)$, and $(a_i(t) = a_i(t-K))$,
  - Then $q_i(t) = 1 \text{ w.p. } 1$
- Else
  - $q_i(t) = \begin{cases} 1 - U_i \left( \frac{1}{|A_i|} \sum_{j=t-K+1}^{t} r_i(j) \right) & \text{w.p. } \epsilon \\ 0 & \text{w.p. } 1 - \epsilon \end{cases}$
Remark 2: In G-NUM, we restrict the rate region to those points achievable by a sequence of $K$ actions. Thereby, the problem of maximizing the utility is posed as a combinatorial optimization problem. A similar approach is seen in [27] to achieve efficient correlated equilibrium.

A. Performance Analysis of G-NUM

In this section, we discuss the optimality of G-NUM. We characterize the performance of G-NUM as $t \to \infty$ and $\epsilon \to 0$. To analyse the performance of G-NUM, we use tools from perturbed Markov chains [24], [25]. We first show that G-NUM induces a perturbed Markov chain (perturbed by $\epsilon$). In Theorem 1, we show that the stochastically stable states (see Definition 2) of the Markov chain induced by G-NUM are the set of actions that maximize the sum utility of the nodes. Define $X_i(t)$ as,

$$X_i(t) = (a(t - K + 1), \ldots, a(t), q(t)).$$

$X_i(t)$ corresponds to the actions of all the nodes in the previous $K$ slots and the “satisfaction” variable of the nodes in the current slot $t$. In the following Lemma, we show that $X_i(t)$ is a regular perturbed Markov chain (perturbed by the algorithm parameter $\epsilon$) with a positive stationary distribution.

**Definition 1:** Regular Perturbed Markov Chain: A Markov process $\tilde{X}_i(t)$, with state space $\tilde{\Omega}$ and transition probability $\tilde{P}_i$, is a regular perturbed Markov process (perturbed by $\epsilon$) if the following conditions are satisfied (see [25]).

1. $\forall \epsilon > 0$, $\tilde{X}_i(t)$ is an ergodic Markov Process
2. $\forall x, y \in \tilde{\Omega}, \lim_{t \to 0} \tilde{P}_i(x, y) = \tilde{P}_0(x, y)$
3. $\forall x, y \in \tilde{\Omega}$, if $\tilde{P}_i(x, y) > 0$ for some $\epsilon > 0$, then,

$$0 < \lim_{\epsilon \to 0} \frac{\tilde{P}_i(x, y)}{\epsilon^{r(x,y)}} < \infty.$$

Here, $r(x,y) \geq 0$ is called the resistance of the transition from $x$ to $y$. This definition ensures that the transition probabilities $\tilde{P}_i(x, y)$ approaches $\tilde{P}_0(x, y)$ at an exponential smooth rate.

**Lemma 1:** $X_i(t)$ is a regular perturbed Markov chain (perturbed by $\epsilon$) over the state space $\Omega = (\mathcal{A}^K, \{0, 1\}^N)$ with a positive stationary distribution $\pi_\epsilon$.

**Proof:** See [28, Lemma 1].

The stationary distribution of the Markov chain $X_i(t)$ characterizes the long term average pay-offs of the nodes. We seek to characterize the stationary distribution of the Markov chain $X_i(t)$ for small $\epsilon > 0$. The following definition helps identify states (the action sequences and average pay-offs) that occur a significant fraction of time, especially, for small $\epsilon$.

**Definition 2:** A state $x \in \tilde{\Omega}$ of a regular perturbed Markov chain $\tilde{X}_i(t)$ with stationary distribution $\pi_\epsilon$ is said to be stochastically stable, if $\lim_{\epsilon \to 0} \pi_\epsilon(x) > 0$.

The following theorem characterizes the stochastically stable states of the Markov chain $X_i(t)$.

**Theorem 1:** Under Assumption 1, the stochastically stable states of the Markov chain $X_i(t)$ are the set of states that optimize the following formulation:

$$\max \sum_{i=1}^N U_i(\tilde{r}_i)$$

where, $\tilde{r}_i = \sum_{a \in A} p(a) f_i(a)$,

$$s.t. a = (a_1, a_2, \ldots, a_N) \in A,$$

$p(a) \in \left\{0, \frac{1}{K}, \frac{2}{K}, \ldots, 1\right\}, \sum_a p(a) \leq 1.$

Further, for a fixed $\epsilon$, the stationary distribution of the optimal states $\pi_\epsilon^{opt}$ satisfies,

$$\pi_\epsilon^{opt} \geq \frac{\alpha_1 \epsilon^{-U_{max}} + \alpha_2 \epsilon^{-U_{2}}}{\alpha_1 \epsilon^{-U_{max}} + \alpha_2 \epsilon^{-U_{2}}}.$$

where $U_{max}$ and $U_2$ are the maximum and second maximum sum utility (with constraints as in (3)) respectively. Also, $\alpha_1 = \Theta(1)$ and $\alpha_2 = \Theta(|\mathcal{A}|^{K^2 N} - 1)$.

**Proof:** (Theorem 1) See Appendix A. \hfill $\square$

**Remark 3:** The fact that $\alpha_2 = \Theta(|\mathcal{A}|^{K^2 N} - 1)$ in (4) is not surprising since in G-NUM we are solving a combinatorial optimization with $|\mathcal{A}|^{K^2 N}$ possible combinations.

**Remark 4:** In Theorem 1, we prove that GNUM optimizes the formulation in (3), where $p(a)$ is restricted to the set $\{0, \frac{1}{K}, \frac{2}{K}, \ldots, \frac{K}{K}\}$. For large $K$, and bounded utility functions, we note that the performance of our proposed algorithm would be approximately optimal (even for small enough $\epsilon > 0$).

**Remark 5:** Markov Chain Monte Carlo (MCMC) based sampling methods have been used to solve combinatorial optimization problems in a distributed way [12], [13]. For constructing MCMC algorithms in a distributed way, the network is required to have a Markov random field structure. In our approach, we do not require the network to have a Markov random field structure.

IV. DISTRIBUTED SUBGRADIENT ALGORITHM FOR CONCAVE UTILITY MAXIMIZATION

In Remark 5, we stated an important relation between G-NUM and MCMC based algorithms. MCMC based utility maximization algorithms were presented in [14], [15], [17], and [18], where the parameters of the MCMC algorithm are adapted to achieve utility maximization for concave utilities. Inspired by these works, we propose a completely uncoupled subgradient algorithm for concave utility maximization.

Throughout this section, we assume that, for all $i$, $U_i$ is increasing and strictly concave with $U_i'(0) < V$. We present Algorithm 2, which we call Concave Network Utility Maximization (C-NUM). As before, every node $i$ is given an internal satisfaction variable $q_i(t)$ taking values 0 and 1. The satisfaction variable $q_i$ serves a similar purpose here, where $q_i = 1$ corresponds to node $i$ being “content” with the current action chosen and $q_i = 0$ represents “discontent” state. The key difference here as compared to G-NUM is that each node is given a weight $\lambda_i(t)$ taking values in $\mathbb{R}_+$. We divide time into frames of $T$ slots each. The first frame contains slots 1 to $T$, second frame contains slots $T + 1$ to $2T$ and so on. The weight $\lambda_i(t)$ is updated at the end of
every frame and is constant during the frame, i.e., $\lambda_i(t) = \lambda_i((l-1)T + 1)$, $t \in [(l-1)T + 1, lT]$. However, a node updates its action $a_i(t)$ and satisfaction variable $q_i(t)$ in every time slot. We shall index frames by $l$ and time slots by $t$. With a slight abuse of notation, we shall denote $\lambda_i(t)$ as $\lambda_i$ at the beginning of frame $l$. Let $\lambda = (\lambda_1, \ldots, \lambda_N)$ denote the vector of weights of the nodes.

The update of $a_i(t), q_i(t)$ at time $t$ requires only the knowledge of the immediate history, i.e., $a_i(t-1), q_i(t-1)$ and $\lambda_i(t)$. In contrast, G-NUM requires the knowledge of previous $K$ actions. At the beginning of a slot $t$, nodes choose their action depending on the satisfaction variable $q_i(t-1)$. If node $i$ was content in time slot $t-1$ i.e., $q_i(t-1) = 1$, then it repeats the action it chose in the previous slot i.e., $a_i(t-1)$ with a large probability $1 - \epsilon^c$. Any other action is chosen uniformly at random with a probability $\epsilon^c$. In case node $i$ was discontent, i.e., $q_i(t-1) = 0$, it chooses an action uniformly at random from $A_i$. Depending on the action profile $a(t)$ chosen, node $i$ receives a pay-off $r_i(t) = f_i(a(t))$. Node $i$ updates its satisfaction variable $q_i(t)$ based on the pay-off it received during slot $t$ and its weight $\lambda_i(t)$. In slot $t$, node $i$ remains content with probability 1 if it was content in the previous slot, repeats its previous action and the pay-off remains unchanged, i.e., $q_i(t-1) = 1, a_i(t) = a_i(t-1)$ and $r_i(t) = r_i(t-1)$. Else, a node becomes content with a small probability $\epsilon^c \frac{\lambda_i(t)r_i(t)}{\lambda_{\text{max}}(t)}$ and remains discontent with probability $1 - \epsilon^c \frac{\lambda_i(t)r_i(t)}{\lambda_{\text{max}}(t)}$. Here, $\lambda_{\text{max}}$ is a positive constant chosen to be greater than $V + 1$, i.e., $\lambda_{\text{max}} > V + 1$. Let $\tilde{s}_i(l)$ be the average pay-off received by node $i$ in frame $l$, i.e.,

$$\tilde{s}_i(l) = \frac{1}{T} \sum_{t=(l-1)T+1}^{lT} r_i(t).$$

Finally, at the end of frame $l$, $\lambda_i(l+1)$ is updated by an approximate sub gradient algorithm given by,

$$\tilde{r}_i(l) = \arg \max_{\alpha \in [0,1]} U_i(\alpha) - \alpha \lambda_i(l),$$

$$\lambda_i(l+1) = [\lambda_i(l) + b(l) (\tilde{r}_i(l) - \tilde{s}_i(l))]^+, \quad (5)$$

where $[x]^+ = \max \{x, 0\}$. Here, $\lambda_i(l+1)$ will serve as the weight for the nodes during frame $l+1$, i.e., from slot $(lT+1)$ to $(l+1)T$.

**Assumption 2:** Assume that $U_i'(0) < V$.

This assumption ensures that the weights $\lambda$ are bounded, which is given by the lemma below. A similar assumption is seen in [17, Lemma 19].

**Lemma 2:** If $\lambda_i(0) < V + 1 \forall i$, then under Assumption 2 we have, $\lambda_i(l) \leq V + 1 \forall l$.

**Proof:** We discuss the proof in Appendix B. □

### A. Performance Analysis of C-NUM

In this subsection, we shall discuss the performance of C-NUM. We first motivate the basis of C-NUM by formulating the dual problem and showing that (5) is the approximate sub-gradient update of the dual problem. In Lemma 3, we derive upper bounds on the mixing time of the Markov chain induced by C-NUM for a constant $\lambda$. This aids in choosing an appropriate frame size. Finally in Theorem 2, we show that C-NUM is optimal.

Consider the optimization problem,

$$\max \sum_i U_i(\tilde{r}_i)$$

s.t. $\tilde{r}_i \leq \sum_a p(a)r_i(a)$,

$$\sum_a p(a) = 1, \quad p(a) \geq 0. \quad (6)$$

The partial Lagrangian of (6) is given by,

$$L(\tilde{r}, p, \lambda) = \sum_i U_i(\tilde{r}_i) - \sum_i \lambda_i(\tilde{r}_i - \sum_a p(a)r_i(a)),$$

$$= \sum_i (U_i(\tilde{r}_i) - \lambda_i\tilde{r}_i) + \sum_a p(a) \sum_i \lambda_i r_i(a),$$

where $\tilde{r} = (\tilde{r}_1, \ldots, \tilde{r}_N)$. The dual of the problem (6) is

$$d(\lambda) = \max_{\tilde{r}, p} L(\tilde{r}, p, \lambda)$$

s.t. $\sum_a p(a) = 1, \quad p(a) \geq 0. \quad (7)$

The maximization in (7) could be split into:

1) Maximization over $\tilde{r}$,

$$\bar{r}(\lambda) = \arg \max_{\alpha \in [0,1]} \sum_i U_i(\alpha_i) - \lambda_i\alpha_i. \quad (8)$$

2) Maximization over $p$,

$$p(\lambda) = \arg \max_{\mu} \sum_a \mu(a) \sum_i \lambda_i r_i(a)$$

s.t. $\sum_a \mu(a) = 1, \quad \mu(a) \geq 0. \quad (9)$
Here, (9) is interpreted as the max weight problem and $\lambda_i$ is interpreted as virtual queue at node $i$. By strong duality, the problem in (6) is equal to the following

$$
\min_{\lambda} \frac{d(\lambda)}{\lambda} \quad \text{s.t.} \quad \lambda_i \geq 0.
$$

(10)

The subgradient of the above dual problem is given by,

$$
\sum_a p(a) \bar{r}_i(a) - \bar{r}_i(l),
$$

where $p, \bar{r}$ are the primal optimal solutions.

**Remark 6:** The central idea is to use the algorithm in [5] (which is the same as G-NUM with $K = 1$) to solve the above maximization over $p$. Recall that G-NUM induces a Markov chain and Theorem 1 characterizes its stationary distribution as $\epsilon \to 0$. However, we need to take care of the following,

1) the time taken to converge to its stationary distribution
2) the effect of using a finite $\epsilon$ in the subgradient algorithm

In C-NUM, we use subgradient method to solve the dual problem in (10), where the dual parameters are updated at the end of each frame. For instance, at the end of frame $l$, the approximate subgradient at node $i$ is given by,

$$
\bar{s}_i(l) - \bar{r}_i(l),
$$

(11)

where $\bar{s}_i(l)$ is the average pay-off obtained in frame $l$ and $\bar{r}_i(l) = \bar{r}_i(\lambda(l))$ solves (8). We omit the dependence on $\lambda(l)$ for ease of notation. In each frame, where $\lambda$ is kept constant, the algorithm induces an ergodic Markov chain, denoted by $Y_e$. This Markov chain maximizes (9) in the limit $\epsilon \to 0$. By using (11) in place of the exact gradient, we need to choose a frame size large enough to ensure that the Markov chain $Y_e$ is close to stationarity and $\epsilon$ small enough that (11) is close to the exact gradient.

**Remark 7:** In [17], Gibbs sampling is used to solve (9), where the Markov random field nature of the setup allows a distributed implementation. In our setup, usage of such reversible Markov Chain Monte Carlo techniques to solve (9) does not lead to a distributed implementation. Another key difference here as compared to [17], is the absence of variational characterization (See Lemma 21 in [17]). Such characterization is possible in [17], due to the structure of the stationary distribution of the Gibbs sampling algorithm. In C-NUM, we do not have such nice structure in the stationary distribution. In fact, we only characterize the stationary distribution as $\epsilon \to 0$. This leads to an approximate subgradient algorithm even if the Markov chain is assumed to converge instantaneously.

In the following discussion, we derive an upper bound for the mixing time of $Y_e$ for a fixed $\lambda$. As we shall see, this bound provides a trade off between $\epsilon$ and frame size $T$.

Let $\pi_\epsilon$ be the stationary distribution of a Markov chain $Y_e$ and $\pi^\epsilon$ be the distribution at time $t$ with $x$ as the initial state, i.e., $\pi_0(x) = 1$. Define mixing time of $Y_e$ as,

$$
\tau(\zeta) = \min \{ t : \sup_x d_V(\pi^\epsilon_t, \pi_\epsilon) < \zeta \},
$$

where $d_V(\pi_\epsilon, \pi_e) = \frac{1}{2} \sum_y |\pi_e(y) - \pi_\epsilon(y)|$ is the total variation distance between $\pi_\epsilon$ and $\pi_e$. Now, we have the following lemma on the mixing time of $Y_e$ for any $\lambda$.

**Lemma 3:**
1) For C-NUM with any fixed $\lambda$, the mixing time of the Markov chain $Y_e(t)$ has the following upper bound.

$$
\tau(\zeta) < \frac{\log(\zeta)}{L(1+1)NK}.
$$

2) For G-NUM, the mixing time is upper bounded as,

$$
\tau(\zeta) \leq \left[ \frac{\log(\zeta)}{c(c+1)} \right] K.
$$

**Proof:** See Appendix C.

**Remark 8:** A small $\epsilon$ would provide a better approximation of (9), however this would lead to a large mixing time and in turn a large frame size.

**Remark 9:** Lemma 3 suggests that, for G-NUM, the mixing time grows exponentially with $K$. This validates the usage of C-NUM against G-NUM for concave utilities, since in G-NUM we increase $K$ for a larger rate region.

Let $\delta(\lambda(l)) = \sum_a \sum (p(\lambda(l)) - \pi_\epsilon(l) \lambda(l)) \lambda(\lambda(l))$, where $p(\lambda(l))$ is the solution to (9) and $\pi_\epsilon(l)$ is the stationary distribution of $Y_e$ with weight $\lambda(l)$. Now we shall state our main result.

**Theorem 2:** Assume $U_i$'s are increasing, strictly concave functions satisfying Assumptions 1 and 2. Then, for fixed frame size $T = \frac{\sum_i U_i}{\eta + \|\nu\|_1}$, we have the following,

1) For step sizes satisfying,

$$
\sum_l b(l) = \infty, \quad \text{and} \quad \sum_l b^2(l) < \infty.
$$

We have,

$$
\lim_{L \to \infty} \inf \sum_i U_i(\hat{r}_i(L)) \geq \sum_i U_i(\hat{r}_i^*) - \lim_{L \to \infty} \hat{b}(L) - \eta,
$$

and

$$
\lim_{L \to \infty} \hat{r}_i(L) - \hat{s}_i(L) = 0,
$$

where $\hat{b}(L), \hat{r}_i(L), \text{and} \hat{s}_i(L)$ are given by,

$$
\hat{b}(L) = \sum_{l=0}^{L-1} b(l), \quad \hat{b}(L) = \frac{1}{b(L)} \sum_{l=0}^{L-1} b(l) \delta(\lambda(l)),
$$

$$
\hat{r}_i(L) = \frac{1}{b(L)} \sum_{l=0}^{L-1} b(l) \bar{r}_i(l), \quad \hat{s}_i(L) = \frac{1}{b(L)} \sum_{l=0}^{L-1} b(l) \bar{s}_i(l).
$$

2) For a fixed step size, $b(t) = b \forall t$,

$$
\lim_{L \to \infty} \inf \sum_i U_i \left( \frac{1}{L} \sum_{l=0}^{L-1} \bar{r}_i(l) \right)
$$

$$
\geq \sum_i U_i(\bar{r}^*) - \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} \delta(\lambda(l)) - \eta - \frac{N b}{2},
$$
and \( \bar{r}_i(l) - \bar{s}_i(l) \leq \frac{V+1}{b} \forall l \), where \( \bar{r}^* \) is the solution to the following optimization problem,

\[
\max \sum_i U_i(\bar{r}_i) \\
\text{s.t. } \bar{r}_i \leq \sum_a p(a)r_i(a), \\
\sum_a p(a) = 1, \ p(a) \geq 0.
\] (12)

Proof: See appendix D.

V. NUMERICAL EXAMPLES

A. Illustration

In this subsection, we illustrate some aspects of C-NUM. We consider an example with two nodes. Each node has two actions to choose from, namely \( a_1 \) and \( a_2 \). The pay-off table is shown in Figure 1. When nodes choose different actions, the node choosing action \( a_2 \) gets a higher pay-off. However the pay-offs are asymmetric, i.e., when node 2 chooses \( a_2 \), it gets a pay-off of 0.8 and when node 1 chooses \( a_2 \), it gets a pay-off of 1 (provided the other node chooses \( a_1 \)). In this example, we aim to maximize the sum of log utility, \( \sum_i \log(1 + \bar{r}_i) \).

1) Effect of Frame Length \( T \) and Number of Iterations \( L \):
In this subsection, we study C-NUM for different frame lengths and iterations of the sub-gradient algorithm. We choose \( \epsilon = 0.01 \). We run C-NUM for 200 and \( 10^6 \) frames. Figures 2 and 3 show the utility of nodes 1 and 2 respectively, where C-NUM is run for 200 frames, with frame lengths \( 10^4 \), \( 10^6 \) and \( 10^7 \) slots. For reference, we plot max weight and Gibbs sampling based algorithms (requires complete information).

In Figure 4, we plot the utility of nodes 1 and 2, where C-NUM is run for \( 10^6 \) frames, with frame lengths of 100 and 10000. From the Figures 2, 3 and 4, observe the following,

1) For this example, the approximate gradient algorithm converges.
2) For a smaller number of frames (200) (See Figure 2 and 3), C-NUM converges with frame lengths of \( 10^6 \) and \( 10^7 \). Also the utilities are close to that got by the Gibbs sampling algorithm.
3) For a larger number of frames, i.e., \( 10^6 \), even for smaller frame length of 100 slots, the algorithm converges. This is explained by stochastic approximation [16].

B. Example Scenarios

In this subsection, we will illustrate G-NUM and C-NUM for two applications namely User Association and Channel Selection in WiFi networks.

1) User Association: Here, nodes (players) correspond to the users and actions correspond to the set of Access Points (APs). The pay-off of user \( i \) corresponds to the throughput \( r_i(t) \). We consider a fixed IEEE 802.11ac WiFi network with 2 Access Points (APs) and 7 users. The performance for the network configuration was evaluated using the network simulator ns-3 with the following configuration parameters.

2) Effect of \( \epsilon \): In this subsection, we study the effect of \( \epsilon \) on the performance of C-NUM. We run C-NUM for \( \epsilon = 0.1, 0.01, 0.001 \) and 0.0001. We fix the frame size as \( 10^6 \) slots and run the algorithm for 200 frames. In Figure 5, we plot the sum utility for different values of \( \epsilon \). We observe that, as \( \epsilon \) is decreased from 0.1 to 0.01, there is a significant increase in sum utility. However, from \( \epsilon = 0.01 \) to 0.001, there is only a marginal increase in the utility. As \( \epsilon \) is further reduced to 0.0001, the algorithm doesn’t converge for the chosen frame size and iteration duration.
The APs are placed 50 meters apart from each other. The UEs are dropped uniformly around the APs over a square of 50 meters. Each user could associate to either of the two APs. We let the APs operate in orthogonal 20 MHz Channels with a maximum achievable throughput of 6.5 Mbps. We consider uplink traffic with saturated queues. We maximize the sum utility function \( \sum_i \log(\delta + \bar{r}_i) \), where \( \bar{r}_i \) is the average throughput of user \( i \). Log utility achieves a proportional fair solution [8]. Since log utility is unbounded, we use \( \log(\delta + \bar{r}_i) \) (\( \delta > 0 \)). We plot the normalized sum utility of the users for G-NUM and C-NUM in Figure 6 with \( \epsilon = 0.2 \). We also plot the max weight based algorithm [9] for reference. We observe that the sum-utility of C-NUM and G-NUM is around 0.42. The performance is close to the max weight based algorithm [9] which achieves a utility around 0.45.

C. Channel Selection

In the channel selection problem, nodes (players) correspond to ad hoc WiFi links (transmitter-receiver pair) and actions correspond to the set of channels that each link could operate. The pay-off for link \( i \) is the throughput \( r_i(t) \) it receives. For the channel selection example, we consider 5 WiFi links (transmitter-receiver pairs) dropped uniformly in an area of 100 square meters. We assume that each link could operate in one of the three 20 MHz channels. We fix the network utility as \( \sum_i \log(\delta + \bar{r}_i) \). We plot the normalized network utility for G-NUM and C-NUM in Figure 7, with \( \epsilon = 0.1 \). We also plot the max weight based algorithm [9] for reference. We observe that C-NUM achieves a utility of 0.71 and G-NUM achieves a utility of 0.69. The performance is very close to the max weight based algorithm, which achieves a utility of 0.73. In comparison to the previous user association example in Figure 6, C-NUM and G-NUM performs better here because \( \epsilon \) was fixed at 0.2 in the user association example, whereas here it is 0.1.

1) Key Observations:

i) We see that G-NUM and C-NUM perform close to the max weight based algorithm [9]. However, there is a small difference in the performance due to finite \( \epsilon \).
TABLE I

| Algorithm                      | Network Model          | Utility                  | Mixing time                                           |
|-------------------------------|------------------------|--------------------------|------------------------------------------------------|
| G-NUM                         | Interdependence        | General                  | \[
\frac{\log(\frac{c}{\epsilon})}{K}\epsilon^{(c-1)/KN}\]   |
| C-NUM                         | Interdependence        | Concave                  | \[
\frac{\log(\frac{c}{\epsilon})}{\epsilon^{(c-1)/N}}\]   |
| Optimal CSMA [17]             | Conflict Graph         | Concave                  | \[
\log\left(\frac{c}{\epsilon}\right)\exp(O(\beta N V))\] |
| Parallel Glauber Dynamics [31]| Conflict Graph with max degree Δ | Stabilizes any arrival with rate < \frac{1}{\Delta-1} | O(log N) |
| Pareto Optimality Through Distributed Learning [5] | Interdependence        | Sum-Rate Maximization    | \[
\frac{\log(\frac{c}{\epsilon})}{\epsilon^{(c-1)/N}}\]   |

ii) Additionally, we observe that C-NUM outperforms G-NUM for the same c. This is explained using the bound in (4). Note that in (4) the denominator scales with K. Thus, the stationary distribution is close to the optimum for C-NUM as compared to G-NUM unless the set of optimal points scale with K.

VI. SUMMARY AND COMPARISONS

In this section, we will summarize G-NUM and C-NUM. G-NUM has the advantage that it maximizes general utilities (not necessarily concave). However, we have seen in Lemma 3 that the mixing time of G-NUM grows exponentially in K (for C-NUM, K = 1). Also, we have seen through simulations that for a fixed c, C-NUM performs better than G-NUM.

Table I provides a comparison of G-NUM and C-NUM with other distributed algorithms. In Remark 1, we stated that the difference between the algorithms in [17] and our model is the assumption on the network. The work in [17] assumes a conflict graph model, whereas we assume interdependence. An important question here is if conflict graph imply interdependence. The answer is negative. To see this, we consider a linear network with three links where i) links 1 and 2 conflict and ii) links 2 and 3 conflict. In this case, when link 1 is transmitting and link 2 is idle, no action change by link 3 could change the service rates of links 1 and 2. This implies that interdependence is not satisfied here. However, this is not a drawback, since significant change in interference can be used as a signal to indicate a change in action.

An important issue with the max weight algorithm is that it has exponential time complexity (number of steps to solve one instance of the problem). In contrast, time complexity of both G-NUM and C-NUM is O(1). This is the case with other distributed algorithms in [5] and [17].

VII. CONCLUSIONS

In this work, we presented two distributed algorithms for utility maximization. This allows a fair allocation of resources, which prior works [4], [5] in this setup have ignored.

In the first algorithm, namely G-NUM, we discretize the rate region, thereby allowing it to be applied for general (possibly non-concave) utilities. We show that the set of achievable points in the rate region could be increased by increasing the parameter K. However, the memory at each node increases as K is increased. The state space of the Markov chain increases with K; and in Lemma 3, we show that the Mixing time upper bound grows exponentially with K.

In the second part, we present another algorithm, C-NUM, which is a sub-gradient algorithm for concave utilities. In comparison to G-NUM, C-NUM only keeps track of the immediate history and the time average service rate. We show convergence in Cesaro averages for decreasing step sizes and time average convergence for fixed step size. Through C-NUM, we show an interesting relationship between completely uncoupled algorithms and Gibbs sampling based utility maximization algorithms. In future, we would like to compare these algorithms for specific network models of interest.

APPENDIX

A. Proof of Theorem 1

To prove theorem 1, we need to characterize the stationary distribution of \( X_\epsilon(t) \) for small \( \epsilon \). For such a characterization, we will use the theory on perturbed Markov chains [24], [25]. Let \( P_\epsilon(x, y) \) denote the transition probability of \( X_\epsilon \) from state \( x \) to state \( y \). Consider the directed graph \( G = (V_G, E_G) \) with the states of the Markov chain as vertices \( V_G \), and an edge \( e_{x,y} \) (\( e \in E_G \)) from vertex \( x \) to vertex \( y \) if and only if \( P_\epsilon(x, y) > 0 \).

A spanning tree rooted at vertex \( x \), \( T_x \) is a subgraph of \( G \) with a path from every vertex \( y \) (\( y \neq x \)) to \( x \), without containing cycles. Note that \( T_x \) is a spanning tree of \( G \) with no outgoing edge from the vertex \( x \). Let the set of all trees rooted at state \( x \) be denoted as \( T_x \). We need the following additional definitions from the theory of regular perturbed Markov processes [25].

1) Let \( x, y \in \Omega \). If \( P_\epsilon(x, y) > 0 \) for some \( \epsilon > 0 \), then,
\[
0 < \lim_{\epsilon \to 0} P_\epsilon(x, y) < \infty,
\]
Here, \( r(x, y) \geq 0 \) is defined as the resistance of the transition from \( x \) to \( y \) (See Definition 1).

2) Consider a sequence of transitions (or a path) \( P = x_1 \to \cdots \to x_k \). The resistance of the path \( r(P) \) is defined as
the sum of the resistances of the one-step transitions in the path, i.e., $r(P) = r(x_1, x_2) + \cdots + r(x_{k-1}, x_k)$.

3) The resistance from vertex $x$ to vertex $y$, $\rho(x, y)$ is the minimum resistance over all paths from $x$ to $y$.

4) The resistance of a tree $T_x$ rooted at $x$, $\rho(T_x)$ is the sum of resistances of the edges in the tree.

5) Stochastic potential of a state $x$, denoted by $\gamma(x)$, is the minimum resistance over all trees rooted at $x$.

Let $\mathcal{T}(a)$ be the set of all spanning trees rooted at state $x_a = (a, q)$ with action profile sequence $a = (a(1), \ldots, a(K))$, where $a(i) \in A$ and $q \in \{0, 1\}^K$ i.e.,

$$\mathcal{T}(a) = \{ T_{(a, q)} | q \in \{0, 1\}^K \}.$$ 

Let $r_{\min}(a)$ denote the minimum resistance over all the trees in $\mathcal{T}(a)$. In the lemma below, we obtain an expression for $r_{\min}(a)$, which will enable us find the stochastically stable states.

**Lemma 4:** The tree with minimum resistance in $\mathcal{T}(a)$ is rooted at state $(a, 1)$, i.e., when all the nodes are content. The minimum resistance $r_{\min}(a)$ is given by $c(\vert A \vert^K - 1) + \sum_{i=0}^{N-1} \prod_{i \neq x} 1 - U_i \left( r_i(a(1)) + \cdots + r_i(a(K)) \right)$.

Further, for any other state $x = (a', q)$, where $q \neq 1$, $r_{\min}(a) < \gamma(x)$, $\forall a, a' \in A^K$.

**Proof:** Denote by $\Omega_1 \subset \Omega$, the set of states with all the nodes content and $\Omega_0 \subset \Omega$, the set of states with all the nodes discontent. Also, let $\Omega_q = \Omega \setminus (\Omega_1 \cup \Omega_0)$. Let $x^1, y^1 \in \Omega_1$, $x^2 \in \Omega_0$ and $x^3 \in \Omega_q$.

We have the following results:

1) For any $x \in \Omega$ and $x \neq x^1$, $\rho(x^1, x) \geq c$. For a transition from $x^1$ to a different state to take place, at least one node should change its action. A content node changes its action with probability $\epsilon$, which has resistance $c$.

2) $\rho(x^1, x^0) = c$. Once a node becomes discontent (which happens with resistance $c$), the other nodes become discontent with zero resistance due to interdependence.

3) Let $x = (a(1), \ldots, a(K), 1) \in \Omega_1$. Then

$$\rho(x^0, x^1) = N - \sum_i U_i \left( f_i(a(1)) + \cdots + f_i(a(K)) \right).$$

The resistance for node $i$ to become content is $1 - U_i \left( f_i(a(1)) + \cdots + f_i(a(K)) \right)$. Since every user must become content from $x^0$, we have the result.

4) $c \leq \rho(x^1, y^1) < 2c$. Here at least one node should change its action which happens with resistance $c$. The upper bound follows from the following, $\rho(x^1, y^1) \leq \rho(x^1, x^0) + \rho(x^0, y^1) = 2c$.

5) $\rho(x^0, x^0) = 0$. In state $x^0$, some nodes are discontent due to interdependence, with zero resistance all the nodes become discontent.

Now, from Lemma 4.3 from [5], we have,

$$\gamma(x^1) = c(\vert A \vert^K - 1) + N - \sum_i U_i \left( r_i(a(1)) + \cdots + r_i(a(K)) \right).$$

(13)

Also we have,

$$\gamma(x^0) = |A|^K c, \forall x^0 \in \Omega_0.$$ 

(14)

(14) follows from 5) and since every outgoing edge from a state $y^1 \in \Omega_1$ has resistance $c$ (there are $|A|^K$ of them).

The stochastic potential of a state $x^q$ is greater than or equal to the stochastic potential of $x^0$, i.e.,

$$\gamma(x^q) \geq |A|^K c, \forall x^q \in \Omega_q.$$ 

(15)

To see (15), let $T_{x^q}$ denote the tree rooted at state $x^q$ with resistance $\gamma(x^q)$. Due to interdependence, we know that there exists a zero resistance path from $x^q$ to $x^0$. Add the zero resistance path to $T_{x^q}$ and remove all the outgoing edges from $x^0$. This gives us a tree rooted at $x^0$ with no additional resistance. This implies $\gamma(x^q) \geq \gamma(x^0)$. Lemma 4 follows from (13), (14), (15) and noting that $c > N$.

The following theorem from [25] completes the proof.

**Theorem 3** [25]: The stochastically stable states of a regular perturbed Markov chain are the states having minimum stochastic potential.

The above theorem insists that the stochastically stable states of the Markov chain $X_t(t)$ are the states where all nodes are content and that minimizes $\gamma(x)$, i.e.,

$$\sum_i 1 - U_i \left( r_i(a(1)) + \cdots + r_i(a(K)) \right).$$

This implies that the stochastically stable states are those that maximize

$$\sum_i U_i \left( r_i(a(1)) + \cdots + r_i(a(K)) \right).$$

The performance of G-NUM for a fixed $\epsilon > 0$ is characterized by the stationary distribution of the non-reversible Markov chain $X_t$. Due to the non-reversible nature of $X_t(t)$, it is difficult to characterize its exact stationary distribution. Hence, we will provide worst case bounds on the stationary distribution of the optimal action sequence. For this purpose, we will use Markov chain tree theorem [24]. By Markov chain tree theorem, the stationary distribution of the Markov chain $X_t$ is given by,

$$\pi(x) = \frac{Q_x}{\sum_{y \in \Omega} Q_y},$$

where, $Q_x = \sum_{T_x \in \mathcal{T}_x} \prod_{(j, k) \in T_x} P_{j, k}$.

We shall compute a lower bound on the stationary probability of the states with least stochastic potential (LSP). Let $x^1_{LSP} \in \Omega_1$ be a state with least stochastic potential. Note that this corresponds to the set of optimal action sequence. We will find an lower bound $Q_{x^1_{LSP}}$. Recall that, for all $i$, $U_i < u_{max} < 1$. Consider a tree rooted at $x^1_{LSP}$ with LSP and denote it as $T_{x^1_{LSP}}$. From Lemma 4, the product of the weights (transition probabilities) of $T_{x^1_{LSP}}$ must necessarily contain the following term,

$$e^{\epsilon(|A|^K - 1) + N} - \sum_i U_i \left( r_i(a(1)) + \cdots + r_i(a(K)) \right),$$

(16)

where the sequence $(a(1), \ldots, a(K))$ maximizes the sum utility. Additionally, the product of the weights of $T_{x^1_{LSP}}$ contain terms with zero resistance. Now, we will find a lower bound on these terms. A transition in $T_{x^1_{LSP}}$ from a state with $k$ content ($k < N$) and $N - k$ discontent nodes takes the following form: $j_1$ of the $k$ nodes become discontent and $j_2$ of the $N - k$ nodes become content with probability at least,

$$\frac{1}{2} \left( 1 - e^\epsilon \right)^k \left( 1 - e^{1 - u_{max}} \right)^{N - k + j_1 - j_2},$$
where \( |A_{\text{max}}| = \max_{i} |A_i| \). In the above, we have not included terms with positive resistance as this has been included in (16). The minimum of the above probability occurs when all \( k \) nodes become discontent i.e., \( j_1 = k \) and \( j_2 = 0 \). This happens with probability at least,

\[
\frac{1}{|A_{\text{max}}|^{N-k}}(1 - e^c)^k (1 - e^{-1-u_{\text{max}}})^N. \tag{17}
\]

Further, the transition from a state with all content nodes has a probability (excluding non-zero resistance terms) at least

\[
\frac{1}{|A_{\text{max}}|^{N-1}}(1 - e^c)^N (1 - e^{-1-u_{\text{max}}})^N. \tag{18}
\]

Let \( Q_{\text{LSP}} \) denote the sum of \( Q_x \)'s with least stochastic potential. From (17), (18) and noting that for each action profile sequence, the number of states with \( k \) content nodes and \( N-k \) discontent nodes is \( NC_k \), we could lower bound \( Q_{\text{LSP}} \) as,

\[
Q_{\text{LSP}} \geq |T_{\text{LSP}}| \epsilon^{\gamma_{\min}} \left( \frac{(1 - e^c)^{N-1}(1 - e^{-1-u_{\text{max}}})^N}{|A_{\text{max}}| - 1} \right) \left( \frac{(1 - e^c)^{N^2-1-N}(1 - e^{-1-u_{\text{max}}})^{N(2^N-1)}}{|A_{\text{max}}|^{N^2-1}} \right) |A|^{K-1},
\]

where \( |T_{\text{LSP}}| \) is the number of trees with LSP. Similarly, for any \( y \in \Omega \)

\[
Q_y \leq |T_{\text{max}}| c^{\gamma_2} \left( \frac{(1 - e^c)^{N-1}(1 - e^{-1-u_{\text{max}}})^N}{|A_{\text{min}}| - 1} \right) \left( \frac{(1 - e^c)^{N^2-1-N}(1 - e^{-1-u_{\text{max}}})^{N(2^N-1)}}{|A_{\text{min}}|^{N^2-1}} \right) |A|^{K},
\]

where \( \gamma_2 = \min_{x,y \neq \gamma(x)} \gamma(x) \) and \( |A_{\text{min}}| = \min_{i} |A_i| \). This corresponds to the state with second maximum utility. Also, \( |T_{\text{max}}| \) is the maximum number of trees rooted at any state respectively. Thus, the stationary distribution of the optimal action sequence \( \pi_{\text{opt}}^x \) is bounded as,

\[
\pi_{\text{opt}}^x \geq \frac{\alpha_1 e^{-U_{\text{max}}} \epsilon^{-N-1}}{\alpha_1 e^{-U_{\text{max}}} + \alpha_2 e^{-N-1}},
\]

where

\[
\alpha_1 = |T_{\text{LSP}}| \left( \frac{(1 - e^c)^{N-1}(1 - e^{-1-u_{\text{max}}})^N}{|A_{\text{max}}| - 1} \right) \left( \frac{(1 - e^c)^{N^2-1-N}(1 - e^{-1-u_{\text{max}}})^{N(2^N-1)}}{|A_{\text{max}}|^{N^2-1}} \right) |A|^{K-1},
\]

and

\[
\alpha_2 = |T_{\text{max}}| \left( 2^N |A|^{K-1} \right) \left( \frac{(1 - e^c)^{N-1}(1 - e^{-1})}{|A_{\text{min}}| - 1} \right) \left( \frac{(1 - e^c)^{N^2-1-N}(1 - e^{-1-u_{\text{max}}})^{N(2^N-1)}}{|A_{\text{min}}|^{N^2-1}} \right) |A|^{K}.
\]

**B. Proof of Lemma 2**

Proof is by induction over \( l \). The statement is true for \( l = 0 \) (by assumption). Now, we assume that \( \lambda_i(l) \in [0, V + 1] \). Consider the two cases:

1. \( \lambda_i(l) \leq V \). In this case, from the update rule, we have

\[
\lambda_i(l + 1) = [\lambda_i(l) + b(l) (\bar{r}_i(l) - \bar{s}_i(l))]^+ \leq \lambda_i(l) + \bar{r}_i(l) \leq V + 1.
\]

2. \( \lambda_i(l) \in (V, V + 1] \). In this case,

\[
\frac{d}{dy}(U_i(y) - \lambda_i y) \leq U'_i(0) - \lambda_i < 0, \quad \forall y.
\]

The steps follow since \( U_i \) is concave and \( \lambda_i < V + 1 \). This implies \( U_i(y) - \lambda_i y \) is a decreasing function in \([0, 1]\); i.e., \( \bar{r}_i = 0 \). \( \Rightarrow \lambda_i(l + 1) \leq V + 1 \).

**C. Proof of Lemma 3**

The Markov chain induced by the algorithm, \( Y_t(t) \) is a non reversible ergodic Markov chain. To analyse the performance of C-NUM, we study the mixing time of the Markov chain \( Y_t(t) \) for a fixed \( \lambda \). We will use Dobrushin’s inequality [32] to derive an upper bound on the mixing time. In this section, we will discuss Dobrushin’s inequality and mixing time based on it. We define ergodic coefficient [32] as,

**Definition 3:** The ergodic coefficient of a transition probability matrix \( P, \delta(P) \) is defined as,

\[
\delta(P) = \frac{1}{2} \sup_{x,y} [p_{xz} - p_{yz}],
\]

where \( d_V \) is the total variation distance.

We shall now state the Dobrushin inequality.

**Theorem 4:** Dobrushin’s Inequality: Let \( P_1 \) and \( P_2 \) be stochastic matrices. Then, \( \delta(P_1 P_2) \leq \delta(P_1) \delta(P_2) \).

**Proof:** See [32, Th.7.1 and Ch. 6].

We will use the above to obtain mixing time bounds.

**Theorem 5:** \( d_V(\mu^T P^t, \nu^T P^t) \leq d_V(\mu, \nu) (\delta(P))^t \).

**Proof:** See [32, Ch. 6 and Th. 7.2].

As a corollary, we have,

\[
d_V(\pi_t, \pi) = d_V(\pi_0^T P^t, \pi^T P^t) \leq d_V(\pi_0, \pi) (\delta(P))^t i, \tag{19}
\]

where, \( \pi_0 \) and \( \pi_t \) are the distribution of the Markov chain at times 0 and \( t \) respectively. The above result indicates that characterizing \( \delta(P) \) shall provide bounds on the convergence.

Now, we shall obtain bounds on the ergodic coefficient \( \delta(P) \) and hence mixing time of the Markov chain \( Y_t \) for a fixed \( \lambda \). The total variation distance is given by,

\[
d_V(p_x, p_y) = 1 - \sum_z p_{xz} \wedge p_{yz}.
\]

Using the above in the definition of ergodic coefficient,

\[
\delta(P) = 1 - \inf_{x,y} \sum_z p_{xz} \wedge p_{yz} \leq 1 - \sum_z p_{\min, z}, \tag{20}
\]

where, \( p_{\min, z} = \min_x p_{xz} \) is the minimum transition probability to state \( z \).

Consider the Markov chain \( Y_t \) in C-NUM with fixed \( \lambda \). Let \( x^1 \) be the a state where all the nodes are content. The minimum
transition probability to \( x^1 \) would correspond to all the \( N \) nodes becoming discontent with probability \( e^{-N} \) and becoming content with probability \( e^{-N} \) (assuming \( \epsilon < 0.5 \)). Thus with a minimum transition probability of \( e^{-N} \).

Also, note that there are \(|A|\) such transitions. Applying the above in (20), we have,

\[
\delta(P) \leq 1 - e^{-N}.
\]

Using (19) from the previous subsection, we have,

\[
d(\pi_t, \pi_s) \leq \left(1 - e^{-N} \right)^t. \tag{21}
\]

From the above, we have,

\[
\tau(\zeta) \leq \frac{\log(1/\zeta)}{e^{-N}}.
\]

### D. Proof of Theorem 2

The proof follows the standard approximate subgradient algorithm in [33], if we assume that the Markov chain \( X_t \) converges to its stationary distribution in (5). We follow the analysis in [17] except that the update in (5) with \( s_t(l) \) replaced by the pay-off averaged over the stationary distribution of \( X_t \) is an approximate subgradient.

Recall, \( \delta(\lambda) = \max_a \sum_i r_i(a) \lambda_i - \sum_a \pi_e(a, \lambda) \sum_i r_i(a) \lambda_i, \) where, \( \pi_e(a, \lambda) \) is the stationary distribution of the Markov chain \( (Y_t) \) and \( s_t(\pi(\lambda)) = \sum_a p(a, \lambda) r_i(a) \) is the service rate obtained with fixed \( \lambda. \) Since,

\[
r_i(l) = \arg \max_{a \in [0,1]} U_t(a) - \alpha_t \lambda_i(l),
\]

we have,

\[
U_t(\tilde{r}_i(l)) - \tilde{r}_i(l) \lambda_i(l) \geq U_t(\hat{r}_i^*) - \hat{r}_i^* \lambda_i(l),
\]

where \( \hat{r}_i^* \) is the optimal solution of (6).

Also, \( \sum_i \hat{r}_i^* \lambda_i(l) \leq \max_a \sum_i r_i(a) \lambda_i(l). \)

Substituting the above in (23) and summing over \( i, \) we get,

\[
\sum_i \tilde{r}_i(l) \lambda_i(l) \leq \sum_i U_t(\tilde{r}_i(l)) - U_t(\hat{r}_i^*) + \max_a \sum_i r_i(a) \lambda_i(l). \tag{24}
\]

From (22) and (24), we have,

\[
2b(l) \lambda_i(l)(\tilde{r}_i - s_t(\lambda(l)))
\]

\[
\leq 2b(l) \sum_i U_t(\tilde{r}_i(l)) - 2b(l)U_t(\hat{r}_i^*) + 2b(l) \delta(\lambda(l)), \tag{25}
\]

where \( s_t(\lambda(l)) = s_t(\pi(\lambda(l))) \) is a slight abuse of notation. Then, for node \( i, \) we have,

\[
\lambda_i^2(l + 1) = \left( \lambda_i(l) + b(l)(\tilde{r}_i(l) - s_t(l)) \right)^2,
\]

\[
\lambda_i^2(l + 1) \leq \lambda_i^2(l) + 2b(l)\lambda_i(l)(\tilde{r}_i(l) - s_t(l)) + b^2(l)(\tilde{r}_i(l) - s_t(l))^2,
\]

\[
\leq \lambda_i^2(l) + 2b(l)\lambda_i(l)(\tilde{r}_i(l) - s_t(l)) + b^2(l). \tag{26}
\]

Summing (26) over all the nodes, we get,

\[
\sum_i \lambda_i^2(l + 1) \leq \sum_i \lambda_i^2(l) + Nb^2(l) + 2b(l)e(l)
\]

\[
+ 2b(l) \sum_i \lambda_i(l)(\tilde{r}_i(l) - s_t(l)), \tag{27}
\]

where, \( e(l) = \sum_i \lambda_i(l)(s_t(l) - \tilde{r}_i(l)) \) is the error due to the fact that the Markov chain \( X_t \) has not converged to its stationary distribution. Substituting (25) in (27), we get,

\[
\sum_i \lambda_i^2(l + 1) \leq \sum_i \lambda_i^2(l) + 2b(l)\lambda_i(l)(\tilde{r}_i(l) - s_t(l)) + \max_a \sum_i r_i(a) \lambda_i(l).
\]

Next we shall consider two cases step sizes of \( b(l). \)

1) **Decreasing Step Size:** Choose \( b(l) \) such that,

\[
\sum_i b(l) = \infty, \text{ and } \sum_i b^2(l) < \infty.
\]

We define the following Cesaro averages,

\[
\bar{b}(L) = \sum_{l=0}^{L} b(l), \quad \bar{U}(L) = \frac{1}{b(L)} \sum_{l=0}^{L-1} \sum_i U_t(\tilde{r}_i(l)),
\]

\[
\bar{\delta}(L) = \frac{1}{b(L)} \sum_{l=0}^{L-1} \sum_i b(l) \delta(\lambda(l)), \quad \bar{r}(L) = \frac{1}{b(L)} \sum_{l=0}^{L-1} \sum_i b(l) \tilde{r}_i(l).
\]

Summing (28) from \( l = 0 \) to \( l = L - 1 \) and normalizing,

\[
\frac{1}{b(L)} \sum_{l=0}^{L-1} \sum_i \lambda_i^2(l + 1)
\]

\[
\leq \frac{1}{b(L)} \sum_{l=0}^{L-1} \sum_i \lambda_i^2(l).
\]
Also by Jensen’s inequality, we have,

\[ b(l)(\bar{r}_i(l) - \bar{s}_i(l)) \leq \lambda_i(l + 1) \leq V + 1, \]

\[ \Rightarrow \lim_{L \to \infty} \frac{1}{b(L)} \sum_{l=0}^{L-1} b(l)(\bar{r}_i(l) - \bar{s}_i(l)) = 0. \]

2) Fixed Step Size \((b(l) = b)\): When \(b(l)\) is a constant,

\[ \sum_i \lambda_i^2(l + 1) \leq \sum_i \lambda_i^2(l) + 2b \sum_i (U_i(\bar{r}_i(l)) - U_i(\bar{r}_i^*)) + 2b\delta(\lambda(l)) + Nb^2 + 2be(l) \text{ (Using (28)).} \]

Averaging from \(l = 0\) to \(L - 1\), we get,

\[ \frac{1}{L} \sum_{l=0}^{L-1} \sum_i \lambda_i^2(l + 1) \]

\[ \leq \frac{1}{L} \sum_{l=0}^{L-1} \sum_i \lambda_i^2(l) + 2b \frac{L-1}{L} \sum_i \delta(\lambda(l)) + Nb^2 \]

\[ + 2b \frac{L-1}{L} \sum_i e(l) + 2b \frac{L-1}{L} \sum_i U_i(\bar{r}_i(l)) - 2b \sum_i U_i(\bar{r}_i^*). \]

\[ \Rightarrow \frac{1}{b(L)} \sum_{l=0}^{L-1} \sum_i \lambda_i^2(l) \leq 0, \]

\[ - \sum_i U_i(\bar{r}_i^*) + \frac{1}{L} \sum_{l=0}^{L-1} \delta(\lambda(l)) + \frac{Nb}{2} + \frac{1}{L} \sum_{l=0}^{L-1} e(l). \]

(32)

Consider, \(\frac{1}{L} \sum_{l=0}^{L-1} e(l) = \frac{1}{L} \sum_{l=0}^{L-1} (e(l) - E(e(l)|F_l) + E(e(l)|F_l)),\) Recall \(S(L) = \sum_{l=0}^{L-1} (e(l) - E(e(l)|F_l))\), with \(b(l) = \frac{1}{L}\). We know that \(S(L)\) converges a.s. By Kronecker’s lemma [34], we have, almost surely,

\[ \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} (e(l) - E(e(l)|F_l)) = 0. \]

Also by our choice of step size, we have, \(E(e(l)|F_l) < \eta\).

Now, using the above and taking limit \(L \to \infty\) in (32),

\[ \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} U_i(\bar{r}_i(l)) \]

\[ \geq \sum_i U_i(\bar{r}_i^*) - \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} \delta(\lambda(l)) - \eta - \frac{Nb}{2}. \]

By Jensen’s inequality, we get,

\[ \lim_{L \to \infty} \sum_i U_i \left( \frac{1}{L} \sum_{l=0}^{L-1} \bar{r}_i(l) \right) \]

\[ \geq \sum_i U_i(\bar{r}_i^*) - \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} \delta(\lambda(l)) - \eta - \frac{Nb}{2}. \]

From the update in (5), for all \(l\), \(\bar{r}_i(l) - \bar{s}_i(l) \leq \frac{V+1}{b}.\)
