Special Issue On The Workshop on MAthematical performance Modeling and Analysis (MAMA 2017)

Mark S. Squillante
Mathematical Sciences Department
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, USA
ms@us.ibm.com

The complexity of computer systems, networks and applications, as well as the advancements in computer technology, continue to grow at a rapid pace. Mathematical analysis, modeling and optimization have been playing, and continue to play, an important role in research studies to investigate fundamental issues and tradeoffs at the core of performance problems in the design and implementation of complex computer systems, networks and applications.

On June 5, 2017, the 19th Workshop on MAthematical performance Modeling and Analysis (MAMA 2017) was held in Urbana-Champaign, IL, USA, sponsored by ACM SIGMETRICS and held in conjunction with SIGMETRICS 2017. This workshop seeks to bring together researchers working on the mathematical, methodological and theoretical aspects of performance analysis, modeling and optimization. It is intended to provide a forum at SIGMETRICS conferences for talks on early research in the more mathematical areas of computer performance analysis. These talks tend to be based on very recent research results (including work in progress) or on new research results that will be otherwise submitted only to a journal (or recently have been submitted to a journal). Thus, part of the goal is to complement and supplement the SIGMETRICS Conference program with such talks without removing any theoretical contributions from the main technical program. Furthermore, we continue to experience the desired result of having short papers from previous MAMA workshops appear as full papers in the main program of subsequent SIGMETRICS and related conferences.

All submissions were reviewed by at least 4 members of the program committee, from which a total of 16 were selected for presentation at the MAMA 2017 workshop. This special issue of Performance Evaluation Review includes (revised) short papers relating to these presentations (arranged in the order of their presentation), which cover a wide range of topics in the area of mathematical performance analysis, modeling and optimization.

The study of Yang et al. considers the control of a Brownian motion with negative drift to minimize a long-run average cost objective with applications to production-inventory systems. Feinberg and Huang study LP formulations, solvable in strongly polynomial time, for certain transient total-cost MDPs and certain average-cost MDPs. The study of Joshi focuses on queueing systems with task replication and seeks to find the fundamental limit of the corresponding increase in service capacity. Aktas et al. consider erasure coding as a more general form of redundancy than simple replication, and analyze the effect of coding on the tradeoff between latency and cost. The study of Lin et al. investigates network design and efficiency loss in open and discriminatory access platforms under networked Cournot competition. Maxey et al. study the price of anarchy and show that in systems with feedback, where the environment may change depending on customer behavior, there can be a benefit of anarchy. The study of Panigrahy et al. considers the fundamental question of when hit-rate based cache utility maximization is more favorable than hit-probability based cache utility maximization. Jiang et al. extend Fagin’s approximation for the LRU cache under IRM to systems where requests for different content form independent stationary and ergodic processes. The study of Goel et al. seeks to provide a theoretical framework for the design of controllers that are decomposed across the different timescales of decentralized and centralized components. London et al. propose a new approach for distributed optimization based on local computation algorithms, an emerging area of theoretical computer science. The study of Aveklouris et al. investigates resource allocation and performance problems within the context of charging electric vehicles. Scully et al. consider optimal job scheduling where each job consists of multiple tasks, each of unknown duration, with precedence constraints between tasks. The study of Baryshnikov and Magnier focuses on random increasing subsequences of random permutations, providing a large deviation principle and an efficient exact algorithm for uniform random sampling. Ganguly et al. analyze a queueing network with moving servers in a large-scale limit where the number of queues goes to infinity while the number of servers converges to some constant. The study of Foss and Stolyar considers a large-scale limit of stationary distributions, as the number of servers goes to infinity, in a parallel server system under Join-Idle-Queue routing. Lu et al. investigate nearly completely decomposable structures in epidemic-like stochastic processes with time-varying behavior.

I sincerely thank each of the speakers and co-authors for their presentation and participation at the MAMA 2017 workshop, as well as each of the other workshop attendees. I also sincerely thank my fellow program committee members: I. Adan, Y. Baryshnikov, S. Borst, G. Casale, C.S. Chang, E. Coffman, J. Dai, E. de Souza e Silva, A. Dieker, N. Duffield, D. Eager, E. Feinberg, D. Gamarnik, A. Ganesh, D. Gaver, R. Gibbens, L. Golubchik, B. Hajek, P. Harrison, A. Jean-Marie, P. Key, G. Latouche, J. van Leeuwaarden, Z. Liu, Y. Lu, J. Liu, M. Mandjes, R. van der Mei, P. Momčilović, P. Nain, D. Nicol, S. Núñez Queija, G. Pang, A. Proutiere, A. Radovanović, K. Ramanan, P. Robert, M. Sharma, E. Smirni, R. Srikant, S. Stoljar, P. Taylor, D. Towsley, B. Van Houdt, N. Walton, A. Wierman, C. Xia, D. Yao, A. Zeevi and B. Zwart. Without all the work and support from both of these groups of people, the MAMA 2017 workshop would not have been possible. I also thank ACM SIGMETRICS for all of their support, and Nidhi Hegde for all of her assistance and support with this special issue of PER.
On the Optimality of Reflection Control, with Production-Inventory Applications

[Extended Abstract]

Jiankui Yang ∗  David D. Yao †  Heng-Qing Ye ‡

ABSTRACT

We study the control of a Brownian motion (BM) with a negative drift, so as to minimize a long-run average cost objective. We show the optimality of a class of reflection controls that prevent the BM from dropping below some negative level r, by cancelling out from time to time part of the negative drift; and this optimality is established for any holding cost function h(x) that is increasing in |x|. Furthermore, we show the optimal reflection level can be derived as the fixed point that equates the long-run average cost to the holding cost. We also show the asymptotic optimality of this reflection control when it is applied to production-inventory systems driven by discrete counting processes.

1. INTRODUCTION

Consider the control of a Brownian motion (BM) with a negative drift, so as to minimize a long-run average cost objective. We show the optimality of a class of reflection controls that prevent the Brownian motion from dropping below some negative level r, by cancelling out from time to time part of the negative drift; and this optimality is established for any holding cost function h(x) that is increasing (i.e., non-decreasing) in |x|, where x is the state variable. This is a natural and desirable form of a cost function, since in applications, the absolute value of the state variable can be interpreted as finished-goods inventory or backordered demand (depending on the sign of x), both incurring costs. To the best of our knowledge, this is the most general form of the cost function for which the optimality of the reflection control is known. (Existing studies in the literature often require second-order properties such as convexity.) Furthermore, let C(r) be the long-run average cost under the reflection control with the level r. We show the optimal reflection level r∗ can be derived as the fixed point that equates the long-run average to the holding cost, C(r∗) = h(r∗).

To prove the optimality of the reflection control, we follow the lower-bound method in Harrison and Taksar [5]. (Also see Harrison [6] and Taksar [9].) Focusing on a sub-class of the class of admissible controls (with the sub-class including the reflection control), we first find a lower bound on the cost objective, and then show this lower bound can be attained by a reflection control with a proper reflection level, which is thus optimal. To connect the result to a discrete production-inventory system driven by counting processes, we use the standard diffusion-limit approach (e.g., Reiman [8]), and establish the asymptotic optimality of the reflection control.

Many related studies in the literature that use BM in production-inventory systems focus on two-sided controls such as the (s, S) policy, whereas the reflection control we focus on here is one-sided. Refer to [1, 3, 4, 7, 10, 11], among many others. However, these papers all need to assume piecewise linear or convex/quasi convex cost functions with polynomially-bounded growth; whereas we only need a cost function h(x) that is increasing in |x|, and we can allow it to have exponentially-bounded growth.

2. THE CONTROL PROBLEM

Given a Brownian motion with a negative drift, X(t) := θt + σB(t), where θ < 0 and σ > 0 are given constants and B(t) denotes the standard Brownian motion, we want to find a control, denoted \( Y(t) \), for all \( t \in [0, +\infty) \), such that the state process

\[
Z(t) = z_0 + X(t) + Y(t), \quad t \in [0, +\infty),
\]

with \( z_0 \) being the initial state, will approach a stationary steady-state distribution for \( F h[Z(+) \infty)] \). In general, the expected long-run average cost is given by

\[
AC(x, Y) = \limsup_{t \to +\infty} \frac{1}{t} \int_0^t h(Z(u))du.
\]

And we have \( AC(z_0, Y) = \mathbb{E}h[Z(+\infty)] \), provided the control \( Y \) induces a steady-state distribution for \( Z(t) \).

• Assumptions on the cost function. Here the cost function \( h(x) \) is assumed to be continuous, and increasing in \( x \geq 0 \) and decreasing in \( x < 0 \). This implies \( h(x) \geq h(0) \) for all \( x \), and \( h(0) \) is assumed to be finite. In both directions, \( x \) tends to \( +\infty \) or \( -\infty \), \( h(x) \) goes to \( +\infty \). Of course, we also need \( \mathbb{E}h[Z(+) \infty)] < +\infty \), and this requires \( h \) to be exponentially bounded; see (7) below.

Note the negative drift of \( X(t) \) will drive the state process to \( -\infty \) without any control, and hence achieve an objective
value that is at one of the two largest extremes of \( h \), i.e., \( h(-\infty) \). Thus, the control \( Y(t) \) is trying to cancel out, from time to time, this negative drift; and in this sense, \( Y(t) \) is a cumulative effort up to \( t \).

- **Admissible controls.** Let \( A \) denote the set of admissible controls. To be admissible a control must be non-anticipative and satisfy the following requirements: \( Y(t) \) is increasing in \( t \in [0, +\infty) \), with \( Y(0) = 0 \).

To motivate, consider a production-inventory system that supplies demand. Suppose demand rate is \( \lambda \) and production rate is \( \mu \). Let the state at time \( t \) be the net demand in the system, i.e., waiting orders minus produced quantities (both are cumulative up to \( t \)). Then, without any control, this net demand is \( (\lambda - \mu)t + \sigma B(t) \), where \( \sigma B(t) \) models the volatility (Gaussian noise) associated with demand (or, with both demand and production). Assume \( \lambda < \mu \); hence, \( \theta := \lambda - \mu < 0 \), and denote this net demand as \( X(t) \). Here, the control is to insert idle time into production; so denote the cumulative idle time up to \( t \) as \( U(t) \). Then, production up \( t \) becomes \( \mu[t - U(t)] \); and, with \( Y(t) = \mu(U(t)) \), the state process can be expressed as follows:

\[
Z(t) = z_0 + (\lambda - \mu)t + \mu U(t) + \sigma B(t)
\]

\[
:= z_0 + X(t) + Y(t).
\]

(3)

Note that \( Z(t) \), when positive, represents the volume of waiting orders; when \( Z(t) \) is negative, its absolute value represents the volume of products waiting to supply demand (i.e., inventory). This also motivates why the cost function \( h(x) \) is increasing in \( x \geq 0 \) and decreasing in \( x \leq 0 \) (the more negative \( x \) is, the higher the cost).

### 3. Reflection Control

Recall, a Brownian motion (starting from 0) with a negative drift will have a stationary limit if it is reflected at some pre-specified value. Hence, we first focus on a sub-class of admissible controls, called “reflection controls,” \( A^* \subset A \); and denote a control in this class as \( Y_\gamma \in A^* \), and denote the corresponding state process as \( Z_\gamma \). The control \( Y_\gamma \) is defined by a reflection level \( r \), meaning it ensures \( Z(t) \geq r \) for all \( t \).

Then, \( Z_\gamma(t) - r \) is a standard reflected Brownian motion (RBM); refer to [2] Section 6.2). It is known that \( Y_\gamma \) and \( Z_\gamma \) can be explicitly expressed as functions of \( X \), the primitive (Brownian motion with drift), as follows:

\[
Y_\gamma(t) = \sup_{0 \leq u \leq t} (r - z_0 - X(u))^+, \quad (4)
\]

\[
Z_\gamma(t) = z_0 + X(t) + \sup_{0 \leq u \leq t} (r - z_0 - X(u))^+. \quad (5)
\]

In addition, complementarity holds: \( |Z_\gamma(t) - r|dY_\gamma(t) = 0 \) for all \( t \), i.e., when \( Z_\gamma(t) > r \), \( Y_\gamma(t) \) cannot increase. Furthermore, the steady-state distribution of \( Z_\gamma(\infty) - r \) follows an exponential distribution with rate \( \gamma := -2\theta/\sigma^2 \) (recall, \( \theta < 0 \)).

Thus, under the reflection control \( Y_\gamma \), we have

\[
Eh(Z_\gamma(\infty)) = \gamma \int_0^{\infty} h(r + x)e^{-\gamma x}dx := C(r). \quad (6)
\]

Note that the left-hand-side is indeed \( AC(z_0, Y_\gamma) \).

Next, we want to find the \( r \) value that minimizes \( Eh(Z_\gamma(\infty)) \). But first note that for this expectation to be finite, we need the function \( h(x) \) to satisfy the following condition: there exist positive numbers \( a \) and \( b < \gamma/2 \) (\( \gamma \) is specified above), such that

\[
h(x) \sim o(ae^{bx}), \quad 3a > 0, \quad 0 < b < \gamma/2. \quad (7)
\]

The \( C(r) \) expression in (6) confirms that the optimal reflection level, if exists, must be negative, since \( C(r) \) is increasing in \( r > 0 \). Taking derivative on \( C(r) \), and applying the variable change, we have

\[
C'(r) = \gamma(C(r) - h(r)).
\]

Hence, the optimal \( r \) can be obtained from

\[
h(r) = C(r) \quad (8)
\]

The optimal solution \( r \) must exist and be strictly negative. To see this, first observe from (6) that \( C(0) > h(0) \). So, the equation in (8) must have a finite and strictly negative solution (denoted as \( r^* < 0 \)), unless \( C(r) > h(r) \) for all \( r \). But then, this means \( C'(r^*) > 0 \), i.e., \( C(r) \) is increasing in \( r < 0 \), which, via (6), contradicts the fact that \( h(r) \) is increasing to \( +\infty \) as \( r \to -\infty \). Furthermore, taking into account \( C(r^*) = h(r^*) \), it is direct to verify \( C(r) \geq C(r^*) \) for \( r \geq r^* \). To summarize, we have

**Proposition 1.** The reflection control \( Y_{r^*} \) is optimal among all controls in the sub-class \( A^* \), with the optimal reflection level \( r^* \) being the solution to \( C(r) = h(r) \).

What remains is to argue that the reflection control \( Y_{r^*} \) is not only optimal within the sub-class \( A^* \) of all reflection controls but also optimal over all admissible controls in \( A \).

To this end, for any admissible control \( Y(t) \), consider another control, \( \tilde{Y}_r(t) := Y_r(t) \wedge Y(t) \). It is then readily verified that (a) \( \tilde{Y}_r(t) \) is an admissible control, and (b) \( \tilde{Y}_r(t) \) yields a lower cost objective than \( Y(t) \), where the reflection level \( r \geq 0 \) is fixed arbitrarily. Thus, it suffices to show (with details spelled out in the full paper):

\[
AC(x, \tilde{Y}_r) \geq AC(x, Y_{r^*}) = C(r^*). \quad (9)
\]

Consequently, we have the following theorem.

**Theorem 2.** The reflection control \( Y_{r^*} \) specified in Proposition 1 is optimal over all controls in the admissible class \( A \), i.e., \( AC(x, Y_{r^*}) \leq AC(x, Y) \) for any initial state \( x \) and any \( Y \in A \).

### 4. Asymptotic Optimality

Consider a discrete version of the production-inventory model outlined in [2], i.e., with both demand and production processes being renewal counting processes. Let \( \{u_i, i = 1, 2, \ldots\} \) denote the inter-arrival times of the orders (demand), an i.i.d. sequence with \( E_u = 1/\lambda \) and the squared coefficient of variation \( c_u^2 \). Let \( \{v_i, i = 1, 2, \ldots\} \) denote the required processing times of the orders, another i.i.d. sequence with \( E_v = 1/\mu \) and the squared coefficient of variation \( c_v^2 \). Assume the two sequences, \( \{u_i, i = 1, 2, \ldots\} \) and \( \{v_i, i = 1, 2, \ldots\} \), are independent; and let \( E(t) \) and \( S(t) \) denote the corresponding counting processes.

Let \( T(t) \) denote the cumulative amount of time production is active (with processing orders) up to time \( t \). Let \( U(t) = t - T(t) \) be the cumulative inactive (idle) time. Let \( Q(t) \) denote the state of the system at time \( t \), the difference between the number of orders that have arrived and
the number of completed products by time \( t \). Then, the dynamics of the system can be written as follows:

\[
Q(t) = Q(0) + E(t) - S(T(t)), \quad t \geq 0.
\]  

(10)

For the above system, reflection control means, whenever the level of inventory reaches a certain level, \( Q(t) = r \), for some negative (integer) \( r \), production will be stopped; i.e.,

\[
T(t) = \int_0^t \mathbb{1}[Q(s) > r] ds.
\]

We want to show that applying reflection control to the above discrete production-inventory system is asymptotically optimal in a precise sense to be spelled below. Consider a sequence of systems as described above, indexed by a superscript \(^{(n)}\), with the \( n \)-th system having arrival rate \( \lambda^{(n)} \), while the service rate \( \mu \) stays fixed. Assume the following limit.

\[
\sqrt{n}(\lambda^{(n)} - \mu) \to 0. \tag{11}
\]

When \( n \to \infty \), the above implies \( \lambda^{(n)} \to \mu \), from below. Thus, when \( n \) is large, the above alludes to a heavily utilized system, with production capacity (\( \mu \)) near saturation. Accordingly, we scale time \( t \) by \( n \) and space by \( 1/\sqrt{n} \) in all processes involved (along with proper centering):

\[
\bar{E}^{(n)}(t) := \frac{1}{\sqrt{n}} E^{(n)}(n t) - \lambda^{(n)} n t, \\
\bar{S}^{(n)}(t) := \frac{1}{\sqrt{n}} S^{(n)}(n t) - \mu n t,
\]

and

\[
\bar{U}^{(n)}(t) := \frac{1}{\sqrt{n}} U^{(n)}(nt), \quad \hat{Q}^{(n)}(t) := \frac{1}{\sqrt{n}} Q^{(n)}(nt).
\]

Then, the dynamics of the \( n \)-th system can be written as,

\[
\bar{Q}^{(n)}(n t) = \hat{Q}^{(n)}(0) + \bar{X}^{(n)}(t) + \bar{Y}^{(n)}(t), \tag{12}
\]

with

\[
\bar{X}^{(n)}(t) = \bar{E}^{(n)}(t) - \bar{S}^{(n)}(t) - \sqrt{n}(\lambda^{(n)} - \mu) t, \\
\bar{Y}^{(n)}(t) = \mu \bar{U}^{(n)}(t), \quad \bar{T}^{(n)}(t) = \frac{n}{\sqrt{n}} \bar{X}^{(n)}(nt).
\]

Applying the standard approach to the diffusion limit of a single-server queue under heavy traffic, we have the following proposition.

**Proposition 3.** Under the condition in (11), along with \( \hat{Q}^{(n)}(0) \to z_0 \), and applying reflection control to the \( n \)-th system with \( \sqrt{nr} \) being the reflection level and \( r \) any given negative integer, we have, as \( n \to \infty \), the following weak convergence (denoted \( \Rightarrow \)):

\[
\bar{X}^{(n)}(t) \Rightarrow X(t) := \sigma B(t) + \theta t, \quad \bar{Y}^{(n)}(t) \Rightarrow Y_r(t); \tag{13}
\]

where \( \sigma^2 = \lambda c_2 + \mu c_2^2 \), \( \theta \) is the constant in (11), and \( Y_r \) is the reflection control in (4). Hence,

\[
\hat{Q}^{(n)}(t) \Rightarrow Z_r(t) := z_0 + X(t) + Y_r(t). \tag{14}
\]

Next, consider any admissible control applied to the \( n \)-th system, under the above time-space scaling. Admissibility means \( \bar{Y}^{(n)}(t) \) must be increasing in \( t \in [0, +\infty) \), with \( \bar{Y}^{(n)}(0) = 0 \), and non-anticipative. Similar to the diffusion limits in the above proposition, we can show that along some subsequence of \( n \), \( \bar{Y}^{(n)}(t) \) will converge to a weak limit, denoted \( Y(t) \), thereby taking the corresponding \( \hat{Q}^{(n)}(t) \) to a weak limit as well, denoted \( Z(t) \), and with

\[
Z(t) := z_0 + X(t) + Y(t),
\]

in parallel to the \((Z_r, Y_r)\) relation in (14). Note, in particular, here \( X(t) \) remains the same as (13), as it involves primitive data only.

Then, from Theorem 2 and Proposition 3, we have

**Theorem 4.** Applying reflection control to the \( n \)-th system as described above, with \( \sqrt{nr} \) being the reflection level and \( r \) specified in Proposition 1, is asymptotically optimal in the sense that its diffusion limit \((as \ n \to \infty) \) yields a long-run average cost \( AC(x, Y_r) \) that is no greater than the long-run average cost of the diffusion limit of the same system under any other admissible control.

5. REFERENCES


Strongly Polynomial Algorithms for Transient and Average-Cost MDPs

Eugene A. Feinberg
Dept. Applied Mathematics and Statistics
Stony Brook University
Stony Brook, NY 11794
eugene.feinberg@stonybrook.edu

Jefferson Huang
School of ORIE
Cornell University
Ithaca, NY 14853
jefferson.huang@cornell.edu

ABSTRACT

This paper considers transient total-cost MDPs with transition rates whose values may be greater than one, and average-cost MDPs satisfying the condition that the expected time to hit a certain state from any initial state and under any stationary policy is bounded above by a constant. Linear programming formulations for such MDPs are provided that are solvable in strongly polynomial time.

Keywords

Markov decision process, algorithm, linear program, transient, total cost, average cost

1. INTRODUCTION

Markov decision processes (MDPs) provide an important framework for the optimization of controlled stochastic systems. For examples of modern applications of MDPs in healthcare, transportation, production systems, communications, and finance, see [2].

It is well-known that there is a close relation between MDPs and linear programming; see e.g., [11]. This relation was used in [19] to develop a combinatorial interior-point algorithm for discounted MDPs and to show, for the first time, that such MDPs can be solved in strongly polynomial time when the discount factor is fixed. This means that the required number of iterations can be bounded above by a polynomial in the number of state-action pairs only. The linear programming formulation of discounted MDPs was again used in [20] to prove that two classic algorithms, the policy iteration method proposed in [10] and the simplex method with Dantzig’s pivoting rule, are also strongly polynomial when the discount factor is fixed. In fact, the complexity estimates for these two algorithms provided in [20] are superior to the estimate for the interior-point algorithm in [19]. Improvements on the complexity estimates in [20] were subsequently provided in [8, 1, 15, 3]. In addition, the estimates for Howard’s policy iteration method were generalized to two-player zero-sum stochastic games in [8, 1], and the analysis in [20] was applied to general linear systems. For examples of modern applications of MDPs in healthcare, transportation, production systems, communications, and finance, see [2].

2. MODEL DESCRIPTION

Consider a discrete-time MDP with finite state set \( X \) and finite action set \( A \). Let \( m := \sum_{x \in X} |A(x)| \) and \( n := |X| \). The one-step costs are denoted by \( c(x, a) \) for \( x \in X \) and \( a \in A(x) \). Finally, to each \( x, y \in X \) and \( a \in A(x) \) is associated a number \( q(y|x,a) \geq 0 \) called the transition rate to \( y \) given that the current state is \( x \) and action \( a \) is performed. For the transient MDP’s considered in this paper, the case where \( \sum_{y \in X} q(y|x,a) > 1 \) for some \( x \in X \) and \( a \in A(x) \) is allowed. Such models are relevant to the control of branching processes; see e.g., [14]. For average-cost MDPs, we will only consider the case where \( q \) is stochastic.
i.e., $\sum_{y \in X} q(y|x, a) = 1$ for all $x \in X$ and $a \in A(x)$, in which case $q(y|x, a)$ is interpreted as the probability that the system transitions to state $y$ given that the current state is $x$ and action $a$ is performed.

A stationary policy is a mapping $\phi : X \rightarrow A$ satisfying $\phi(x) \in A(x)$ for each $x \in X$; let $F$ denote the set of all such policies. It can be shown that it suffices to consider stationary policies for the optimality criteria considered in this paper. Under $\phi \in F$, the decision-maker always selects the action $\phi(x)$ when the current state is $x$. For $\phi \in F$, consider the matrix of one-step transition rates $Q_\phi$ with elements $q(y|x, \phi(x))$, $x, y \in X$. For a matrix $B$ with elements $B(x, y)$ for $x, y \in X$, let $|B| := \max_{x \in X} \sum_{y \in X} |B(x, y)|$.

For undiscounted total costs, which are considered in Section 3, the following transience condition [17] is assumed to hold.

**Assumption T.** The MDP is transient, that is, there is a constant $K \geq 1$ that satisfies $\|\sum_{n=0}^{\infty} Q_\phi^n\| \leq K < \infty$ for all $\phi \in F$.

Assumption T can be checked in strongly polynomial time using the procedure described in [18, proof of Theorem 1].

For $\phi \in F$, let $c_\phi(x) := c(x, \phi(x))$ for $x \in X$. Under Assumption T, the total cost incurred under $\phi \in F$, when the initial state is $x \in X$, is $v^\phi(x) := \sum_{n=0}^{\infty} Q_\phi^n c_\phi(x)$. A policy $\phi$ is total-cost optimal if $v^\phi(x) = \inf_{\phi \in F} v^\phi(x)$ for all $x \in X$. The following characterization of Assumption T [5, Proposition 1] will be used to define the linear programs given in Sections 3 and 4.

**Proposition 1.** Assumption T holds if and only if there is a function $\mu : X \rightarrow [1, \infty)$ that is bounded above by $K$ and satisfies

$$\mu(x) \geq 1 + \sum_{y \in X} q(y|x,a)\mu(y), \quad x \in X, \ a \in A(x).$$

(1)

For average costs, which are dealt with in Section 4, Assumption HT on hitting times formulated below is assumed to hold. To state it, for $z \in X$ and $\phi \in F$ consider the matrix $Q_\phi$ with elements $Q_\phi(x, y) := q(y|x, \phi(x))$ if $x \in X$ and $y \neq z$, and $Q_\phi(x, z) := 0$ for $x \in X$.

**Assumption HT.** There is a state $\ell \in X$ and a constant $K^*$ satisfying $\|\sum_{n=0}^{\infty} i Q_\phi^n\| \leq K^* < \infty$ for all $\phi \in F$.

Assumption HT is equivalent to state $\ell$ being recurrent under all stationary policies, which according to [7] implies that Assumption HT can be checked in strongly polynomial time. We remark that any MDP satisfying Assumption HT is unichain, and that in general the problem of checking if an MDP is unichain is NP-hard [16].

For the initial state $x \in X$, the average cost incurred under $\phi \in F$ is $\omega^\phi(x) := \limsup_{N \rightarrow \infty} \frac{1}{N} \sum_{n=0}^{N-1} Q_\phi^n c_\phi(x)$. A policy $\phi_*$ is average-cost optimal if $\omega^{\phi_*(x)} = \inf_{\phi \in F} \omega^\phi(x)$ for all $x \in X$.

**3. UNDISCOUNTED TOTAL COSTS**

Let $\mu$ be a function satisfying the conditions in Proposition 1. When the constant $K$ in Assumption T is fixed, a total-cost optimal policy can be computed in strongly polynomial time by using the transformation in [5, Sec. 3.1] of the original problem into a discounted MDP, with discount factor $(K - 1)/K$, whose transition rates are stochastic. It follows from [5, Prop. 2] that a total-cost optimal policy for the original transient MDP can be computed by solving the following LP.

Minimize $\sum_{x \in X} \sum_{a \in A(x)} \mu(x)^{-1} c(x, a) z_{x,a}$

such that $\sum_{a \in A(x)} z_{x,a} - \sum_{y \in X} q(x,y,a)\mu(x) z_{y,a} = 1, \quad x \in X,$

$z_{x,a} \geq 0, \ x \in X, \ a \in A(x).$ 

The estimates for discounted MDPs in [15] imply the following complexity estimates for this LP.

**Proposition 2.** The block-pivoting simplex method that corresponds to Howard’s policy iteration algorithm for discounted MDPs needs $O((m - n)K \log K)$ iterations to solve the above LP. In addition, the simplex method with Dantzig’s rule needs at most $O(n(m - n)K \log K)$ iterations to solve the above LP.

We remark that the above estimate for Howard’s policy iteration algorithm matches the one in [3] for transient MDPs, which was obtained without reducing the original problem to a discounted one. In fact, it follows from the definition of the transformation in [5, Sec. 3.1] that Howard’s policy iteration for the constructed discounted MDP, which is equivalent to a block-pivoting simplex method for the above LP, corresponds to Howard’s policy iteration for the original transient MDP. On the other hand, it can be shown that applying the simplex method with Dantzig’s rule to the above LP is different than applying this version of the simplex method to the LP formulation for transient MDPs considered in [3]. In addition, observe that, when $K$ is fixed, the above estimate for the simplex method with Dantzig’s rule is better than the one for Dantzig’s rule in [3].

Using the estimates in [3], it can be shown that a suitable function $\mu$ can be computed using $O((m - n)K \log K)$ iterations of Howard’s policy iteration algorithm. Since each iteration of both the simplex and policy iteration methods can be completed using $O(n^2 + mn)$ arithmetic operations, the preceding implies the following theorem.

**Theorem 3.** Suppose the constant $K$ in Assumption T is fixed. Then both the block-pivoting simplex method corresponding to Howard’s policy iteration algorithm for discounted MDPs, as well as the simplex method with Dantzig’s rule, can be used to compute a total-cost optimal policy in strongly polynomial time.

**4. AVERAGE COSTS PER UNIT TIME**

In this section, we assume that the transition rates $q$ are stochastic. According to Proposition 1, there is a function $\mu^* : X \rightarrow [1, \infty)$ that satisfies $\mu^* \leq K^*$ and $\mu^*(x) \geq 1 + \sum_{y \in X\setminus\{x\}} q(y|x,a)\mu^*(y)$ for all $x \in X$ and $a \in A(x)$. When the constant $K^*$ in Assumption HT is fixed, an average-cost optimal policy can be computed in strongly polynomial time by transforming the original problem into a discounted one with discount factor $(K^* - 1)/K^*$ using the transformation in [5, Sec. 4.1]. It follows from [5, Prop. 8] that an average-cost optimal policy for the original transient MDP can be computed...
by solving the following LP.

\[
\begin{align*}
\text{minimize} & \quad \sum_{x \in X} \sum_{a \in A(x)} \mu^*(x)^{-1} c(x, a) z_{x,a} \\
\text{such that} & \quad \sum_{a \in A(x)} z_{x,a} - \sum_{y \in Y} \sum_{a \in A(y)} \frac{q(y|y, a) \mu^*(x)}{\mu^*(y)} z_{y,a} = 1, \quad x \neq \ell, \\
& \quad \sum_{a \in A(\ell)} z_{\ell,a} - \sum_{y \in Y} \sum_{a \in A(y)} \frac{\mu^*(y) - 1 - \sum_{x \in X} q(x|y, a) \mu^*(x)}{\mu^*(y)} z_{y,a} = 1, \\
& \quad z_{x,a} \geq 0, \quad x \in X, \quad a \in A(x).
\end{align*}
\]

The estimates for discounted MDPs in [15] imply the following complexity estimates for this LP.

**Proposition 4.** The block-pivoting simplex method that corresponds to Howard’s policy iteration algorithm for discounted MDPs needs \(O((m-n)K^* \log K^*)\) iterations to solve the above LP. In addition, the simplex method with Dantzig’s rule needs at most \(O(n(m-n)K^* \log K^*)\) iterations to solve the above LP.

The following theorem can then be proven in a way analogous to the case of transient total-cost MDPs.

**Theorem 5.** Suppose the constant \(K^*\) in Assumption HT is fixed. Then both the block-pivoting simplex method corresponding to Howard’s policy iteration algorithm for discounted MDPs, as well as the simplex method with Dantzig’s rule, can be used to compute an average-cost optimal policy in strongly polynomial time.

It follows from the definition of the transformation in [5, Sec. 4.1] that applying Howard’s policy iteration algorithm to the constructed discounted MDP, which is equivalent to a block-pivoting simplex method for the above LP, corresponds to a block-pivoting simplex method for the LP that is typically used to solve unichain average-cost MDPs [11, Sec. 4.6]. On the other hand, this latter LP, even when Assumption HT holds with a constant \(K^*\), may not satisfy the sufficient conditions given in [12] under which the simplex method with Dantzig’s rule is strongly polynomial.

**Acknowledgements.** This research was partially supported by NSF grants CMMI-1335296 and CMMI-1636193.

5. REFERENCES


1. INTRODUCTION
The service capacity (maximum rate of task completion) of a multi-server system is typically the sum of the service rates of individual servers. Several recent works study queueing systems with task replication, where the copies are canceled when any one replica is served. Replication affects the system in two ways: 1) replicas provide load-balancing by finding the shortest among the queues that they join, and 2) redundant time spent by multiple servers on the same task can add load to the system. Contrary to intuition, [1-4] identify cases where replication can in fact reduce the system load, and thus increase service capacity. We seek to find the fundamental limit of this capacity boost, which is an open problem. We present a Markov Decision Process (MDP) framework to find the throughput-optimal replication policy. The MDP is hard to solve in general, and we have to resort to myopic replication policies. To help quantify the gap from optimality, we present an upper bound on the service capacity for the two server case.

2. PROBLEM FORMULATION
Consider a system of $K$ servers with a central queue of tasks, as shown in Fig. 1. Since our objective is to maximize service capacity, we do not explicitly define an arrival process and assume that the queue is never idle.

2.1 Task Service Times
Server $i$ takes time $S = Y X_i$ to finish a task assigned to it. The random variable $X_i$ captures the variability in task service time due to server slowdown. It is independent across servers, and i.i.d. across tasks assigned to any one server. The dependence of the service time on the size of the task is captured by $Y$, which is independent of $X_i$ for all $i$. This method of multiplying the randomness from the two sources of variability was introduced in [5]. We also consider a cancellation delay $\Delta$ at all servers running a replicated task, after which they can serve subsequent tasks.

2.2 Scheduling Policy
When a server becomes idle, the scheduler can take one of two possible actions:
- **new**: assign a new task to that server, or
- **rep**: replicate a task that is currently running at one or more servers.

![Fig. 1: A task replicated at two idle servers 1 and 2 takes time $Y \min(X_1, X_2)$ to finish, where $Y$ captures the task-size variability and $X_i$ captures the server slowdown.](image)

The space of scheduling policies with the new and rep actions is denoted by $\Pi_{n,r}$. The scheduling policy can be based on the distributions of $Y$, $X_1$, ..., $X_K$, but the scheduler does not know their realizations for currently running tasks.

Note that all policies in $\Pi_{n,r}$ are work-conserving, that is, they never allow a server to be idle for a non-zero time interval. In the extended version [6] we show that there is no loss of generality in focusing on work-conserving policies.

2.3 Throughput Metric
Our objective is to determine the policy $\pi_{n,r}$ that maximizes the throughput, which is defined as follows.

**Definition 1 (Throughput $R$).** Let $T_1(\pi) \leq T_2(\pi) \leq \ldots \leq T_n(\pi)$ be the departure times of tasks $1, 2, \ldots, n$, scheduled using policy $\pi$. The throughput is defined as

$$R(\pi) \triangleq \lim_{n \to \infty} \frac{n}{T_n(\pi)}.$$  \hspace{1cm} (1)

We denote the maximum achievable throughput over all policies in $\Pi_{n,r}$ by $R^*_{n,r} = \max_{\pi \in \Pi_{n,r}} R(\pi)$. The policy $\pi^*_{n,r}$ that achieves $R^*_{n,r}$ is called the throughput-optimal policy.

**Claim 1.** For any work-conserving policy, $R = K/E[C]$, where $C$ is the total time spent by the servers per task.

Thus, minimizing $E[C]$ is equivalent to maximizing $R$.

3. FINDING THE OPTIMAL POLICY

3.1 No Replication and Full Replication
First let us compare the throughput of two extreme policies: no replication and full replication.
Lemma 1 (No Replication). If each task is assigned to the first available idle server, the throughput is,

\[ R_{\text{NoRep}} = 1 \sum_{i=1}^{K} \frac{1}{\mathbb{E}[Y] \mathbb{E}[X_i]} \]  

Lemma 2 (Full Replication). Suppose each task is assigned to all servers, and as soon as one replica finishes, the others are canceled. The resulting throughput is,

\[ R_{\text{FullRep}} = \frac{1}{\Delta + \mathbb{E}[Y] \mathbb{E}\left[\min(X_1, X_2, \ldots, X_K)\right]} \]  

The proofs are given in [6]. Using Lemma 1 and Lemma 2 we can compare the two policies for any given \( X_1, \ldots, X_K, Y \) and cancellation delay \( \Delta \).

Example 1. Consider a system with two servers, and assume that the task size variability \( Y = 1 \) and the cancellation delay \( \Delta = 0 \). The service times of the two servers are

\[ X_1 = 2 \]  
\[ X_2 = \begin{cases} 1 & \text{w.p. } 1 - p \\ 20 & \text{w.p. } p \end{cases} \]  

Fig. 2 compares the throughputs with full replication and no replication as \( p \) varies from 0 to 0.5. For \( p > 0.068 \), the FullRep policy outperforms NoRep.

3.2 MDP Formulation of the Optimal Policy

Instead of replicating tasks upfront, replicas could be added conditionally if the original task does not finish in some given time. We now propose a Markov Decision Process (MDP) framework to search for the best replication policy.

3.2.1 State-space

Let us denote the state evolution by \( s_0, s_1, \ldots, s_i, \ldots \) such that the system transitions to state \( s_i \) as soon as the \( i^{th} \) task departs. A state \( s \in [B, t, d] \) where \( B \) contains disjoint sets of server indices that are running the unfinished tasks in the system. For example, if \( B = \{1\}, \{2, 3\} \) there are two unfinished tasks in the system, one running on server 1 and another on servers 2 and 3. The vector \( t = (t_1, t_2, \ldots, t_K) \) where \( t_k \) is the time spent by server \( k \) on its current task. If a server is idle, its \( t_k = 0 \). The \( d \) term ensures that each state transition corresponds to a single task departure. If \( d + 1 \) tasks exit the system simultaneously and result in the task assignment set \( B \) and elapsed-time vector \( t \), then the system goes through states \([B, t, d] \rightarrow [B, t, d - 1] \rightarrow \cdots \rightarrow [B, t, 0]\).

3.2.2 Actions

In states \( s = [B, t, 0] \), the scheduler can assign new tasks to idle servers, or replicate existing tasks. No tasks are assigned in the exit states \( s = [B, t, d] \) with \( d > 0 \). Thus, for these states, the action space \( A_s \) contains a single placeholder null action. The system directly transitions to \([B, t, d - 1]\).

3.2.3 Cost

The cost \( C(s, s', a) \) of taking action \( a \) in state \( s \) and going to state \( s' \) is defined as the total time spent by the servers in that interval. Thus, the throughput-optimal policy is

\[ \pi^*_s = \arg \min_{\pi \in \Pi_{n, r}} \sum_{j=0}^{\infty} C(s_j, s_{j+1}, \pi(s_j)) \]  

Fig. 2: Comparison of the throughputs of different replication policies, for the service times defined in Example 1. The derivation of the upper bound on \( R_{n, r} \) is given in Section 4.

For the service distributions in Example 1, we can solve the MDP. The optimal policy (illustrated in Fig. 2) is to replicate a server 2’s task at server 1 only if it does not finish in 1 second. In general the MDP can have a large state-space even for simple service distributions. And if \( X_i \) for any \( i \) or \( Y \) is a continuous random variable, then the MDP will have a continuous state-space and it is even harder to solve.

3.3 Proposed Replication Policies

As an alternative to solving the MDP, we propose a myopic policy called the MaxRate policy.

Definition 2 (MaxRate Policy). When one or more servers become idle, the MaxRate policy chooses the action \( a \) that maximizes the instantaneous service rate \( R(a) \) which is defined as,

\[ \hat{R}(a) \triangleq \sum_{m=1}^{M(a)} \frac{1}{\mathbb{E}[D_m(a)]}. \]  

where \( M(a) \) is the number of unfinished tasks after taking action \( a \), and \( \mathbb{E}[D_m(a)] \) is the expected remaining time until the departure of task \( m \), assuming it is not replicated further.

Corollary 1. Consider a two server system, with deterministic task size \( Y = 1 \) and no cancellation delay \( \Delta = 0 \). Suppose server 1 becomes idle, and the task running on server 2 has spent time \( t_2 > 0 \) in service. Then MaxRate launches a replica at server 1 if

\[ \frac{1}{\mathbb{E}\left[\min(X_1, X_2^*)\right]} > \frac{1}{\mathbb{E}[X_1]} + \frac{1}{\mathbb{E}[X_2^*]}, \]  

where \( X_2^* = (X_2 - t_2) | X_2 > t_2 \), the residual computing time. Otherwise it assigns a new task to server 1.

Fig. 2 illustrates the MaxRate policy for the service distributions in Example 1. In this case the throughput of the MaxRate policy is the maximum of the throughputs of the NoRep and FullRep policies.

In general, (8) can help find replication thresholds \( t_{i \rightarrow j} \) such that a task running on server \( i \) is replicated at server \( j \) if it does not finish in \( t_{i \rightarrow j} \) seconds. Based on this idea we propose another policy called AdaRep(t), which is directly parametrized by a replication threshold vector \( t \).
Definition 3 (AdaRep Policy). Consider a vector \( \mathbf{u} = (j_1, j_2, \ldots, j_k) \) for \( k < K \) such that a task first launched on server \( j_k \) was later replicated on \( j_2, j_3 \) and so on. Replicate this task at server \( i \) if server \( j_k \) has spent at least \( t_{a\rightarrow i} \) time on it. Otherwise assign a new task to the idle server. If more than one tasks satisfy the replication condition, choose the task whose elapsed time is closest to its \( t_{a\rightarrow i} \).

For example for \( K = 2 \) servers, the vector \( t = [t_{1\rightarrow 2}, t_{2\rightarrow 1}] \). The optimal policy shown in Fig. 2 obtained by solving the MDP is AdaRep([\( \infty, 1 \)]). In the next section we propose a method to choose \( t \) for the two-server case.

4. UPPER BOUND ON CAPACITY

To quantify the optimality gap of a policy without solving the MDP, we need an upper bound on \( R_{n,r}^* \). Recall that in our problem formulation, tasks can be replicated only at time instants when one or more servers become idle. To find an upper bound, we consider that the scheduler is also allowed to pause ongoing tasks.

Definition 4 (The Pause-and-Replicate System). A task can be replicated at any server where it is not already running by pausing the ongoing task on that server. The paused task is resumed after the replica is served or canceled.

The set of feasible policies \( \Pi_{n,r} \) is a subset of \( \Pi_{n,r} \), the set of policies in the pause-and-replicate system. Thus,

\[
R_{p,r}^* = \max_{\pi \in \Pi_{p,r}} R(\pi) \geq \max_{\pi \in \Pi_{n,r}} R(\pi) = R_{n,r}^*.
\]

4.1 Evaluating the Upper Bound

In the pause-and-replicate framework, AdaRep(\( t \)) can replicate a task exactly after time \( t_{a\rightarrow i} \), instead of waiting for server \( i \) to become idle. In Theorem 1 below, we obtain a closed-form expression for the throughput \( R_{p,r}^*(t) \) of the AdaRep policy for \( K = 2 \) servers and \( Y = 1 \). In [6] we show that thus there is no loss of generality in focusing on AdaRep policies. Thus, the upper bound \( R_{n,r}^* = R_{p,r}^*(t^*) \), the throughput of the best AdaRep policy.

Theorem 1. The throughput \( R_{p,r}^*(t) = [t_{1\rightarrow 2}, t_{2\rightarrow 1}] \) of the AdaRep policy in the pause-and-replicate framework can be expressed as follows. For \( t_{1\rightarrow 2} > 0 \) and \( t_{2\rightarrow 1} > 0 \),

\[
R_{p,r}^*(t) = \frac{1}{1 + \gamma_{1\rightarrow 2} + \gamma_{2\rightarrow 1}} \left( \frac{1}{E[X_{1\rightarrow 2}^*(t_{1\rightarrow 2})]} + \frac{1}{E[X_{2\rightarrow 1}^*(t_{2\rightarrow 1})]} \right),
\]

where,

\[
\gamma_{1\rightarrow 2} = \frac{\Pr(X_i > t_{1\rightarrow 2})}{E[X_{1\rightarrow 2}^*(t_{1\rightarrow 2})]} \left( \frac{\Delta + E[\min(X_{1\rightarrow 2}^*(t_{1\rightarrow 2}), X_j)]}{E[X_{1\rightarrow 2}^*(t_{1\rightarrow 2})]} \right),
\]

and \( X_{1\rightarrow 2}^*(\tau) = \min(X_1, \tau) \), the truncated part of \( X_1 \), and \( X_{2\rightarrow 1}^*(\tau) = (X_1 | X_1 > \tau) - \tau \), the residual service time after \( \tau \) seconds of service. If \( t_{1\rightarrow 2} = 0 \) or \( t_{2\rightarrow 1} = 0 \), \( R_{p,r}^*(t) = 1/(\Delta + E[\min(X_1, X_2)]) \).

Proof Sketch. Consider the case \( t_{1\rightarrow 2} > 0 \) and \( t_{2\rightarrow 1} > 0 \). Time can be divided into intervals as illustrated in Fig. 3. In Type 0 intervals, no tasks are replicated. In a Type \( i \) interval, both servers are serving a task that was originally launched on server \( i \). The overall throughput is

\[
R_{p,r} = \mu_0 R_0 + \mu_1 R_1 + \mu_2 R_2.
\]
Effective Straggler Mitigation: Which Clones Should Attack and When?

Mehmet Fatih Aktaş
Rutgers University
mehmet.aktas@rutgers.edu

Pei Peng
Rutgers University
pei.peng@rutgers.edu

Emina Soljanin
Rutgers University
emina.soljanin@rutgers.edu

1. INTRODUCTION AND MODEL

Motivation: Distributed (computing) systems aim to attain scalability through parallel execution of multiple tasks constituting a job. Each of these tasks is run on a separate node, and the job is completed only when the slowest task is finished. It has been observed that task execution times have significant variability, e.g., because of multiple job resource sharing [1]. The slowest tasks that determine the job execution time are known as “stragglers”.

Two common performance metrics for distributed job execution are 1) Latency, measuring the execution time, and 2) Cost, measuring the resource usage. Job execution is desired to be fast and with low cost, but these are conflicting objectives. Replicating tasks and running the replicas over separate nodes has been shown to be effective in mitigating the effect of stragglers on latency [2], and is used in practice [3]. Recent research proposes to delay replication, and clone only the tasks that at some point appear to be straggling, in order to reduce the cost [4].

Erasure coding is a more general form of redundancy than simple replication, and it has been considered for straggler mitigation in both data download [5] and, more recently, in distributed computing context [6]. We here take this line of work further by analyzing the effect of coding on the tradeoff between latency and cost. As in [4], that deals with this issue in the context of replication, we consider systems where coded redundancy is introduced with a delay in order to reduce the cost, and examine the impact of that delay on latency. In [2], introduction of redundancy has been playfully described as attack of the clones. We here examine whether the redundancy should be simple replication or coding and when it should be introduced. That is, following the analogy of [2], we ask which clones should attack and when.

System Model: In our system, a job is split into $k$ tasks. The job execution starts with launching all its $k$ tasks, and the redundancy is introduced only if the job is not completed by some time $\Delta$.

In replicated-redundancy $(k, c, \Delta)$-system, if the job still runs at time $\Delta$, $c$ replicas for each remaining task are launched. In coded-redundancy $(k, n, \Delta)$-system, if the job still runs at time $\Delta$, $n-k$ redundant parity tasks are launched where completion of any $k$ of all launched tasks results in total job completion (see Fig. 1). Note that this assumption does not impose severe restrictions. Any linear computing algorithm can be structured in this way simply by using linear erasure codes. Particular examples can be found in e.g., [6] and references therein.

We assume that task execution times are iid and follow one of the three canonical distributions: 1) $Exp(\mu)$; commonly used to model execution of small-size tasks, 2) $SExp(D, \mu)$; constant $D$ plus $Exp(\mu)$ noise, used when the job size affects the execution time [4], (3) $Pareto(\lambda, \alpha)$; canonical heavy-tail distribution that is observed to fit task execution times in real computing systems [1, 7].

We use $T$ to denote the job execution time. Cost is defined as the sum of the lifetimes of each task involved in job execution. There are two main setups that define cost: 1) Cost with task cancellation $C^c$; remaining outstanding tasks are canceled upon the job completion, which is a viable option for distributed computing with redundancy, 2) Cost without task cancellation $C$; tasks remaining after job completion run until they complete, which, for instance, is the only option for data transmission over multi-path network with redundancy.

In this paper, we analyze the effect of replicated and coded redundancy on cost and latency tradeoff. Specifically, we present exact expressions for expected latency and cost under delayed and zero-delay redundancy schemes. From these expressions, we observe that pain and gain of redundancy are strongly correlated with the tail of task execution time.

Summary of Observations: Coding allows us to increase degree of redundancy with finer steps than replication, which translates into greater achievable cost vs. latency region. Delaying coded-redundancy is not effective to trade off latency for cost, therefore, primarily the degree of redundancy should be tuned for the desired cost and latency. Coding is shown to outperform replication in terms of cost and latency together. When the task execution time has heavy tail, we make a possibly counterintuitive observation that redundancy can reduce cost and latency simultaneously, where the reduction depends on how heavy the tail is.
2. RESULTS AND OBSERVATIONS

We next state expressions for the expected latency and cost under replicated and coded redundancy. Note that these quantities depend on \( c \), the number of tasks the job is split into, the redundancy level (\( c \) in the replicated and \( n \) in the coded systems), as well as \( \Delta \), the time the when the redundancy is introduced.

**Notation:** \( H_n \) is the \( n \)th harmonic number defined for \( n \in \mathbb{Z}^+ \) as \( \sum_{i=1}^{n} \frac{1}{i} \) and for \( n \in \mathbb{R} \) as \( \int_{0}^{n} \frac{1}{x} dx \). Incomplete Beta function \( B(q; m, n) \) is defined for \( q \in [0, 1] \), \( m, n \in \mathbb{R}^+ \) as \( \int_{0}^{q} u^{m-1}(1 - u)^{n-1} du \) and Beta function as \( B(m, n) = B(1; m, n) \). Gamma function \( \Gamma(x) \) is defined as \( \int_{0}^{\infty} u^{x-1} e^{-u} du \) for \( x \in \mathbb{R} \) and as \( (x-1)! \) for \( x \in \mathbb{Z}^+ \).

**Expected Latency and Cost with Replication:**

**Theorem 1.** Under the exponential task execution time \( X \sim \text{Exp}(\mu) \), expected latency in the replication \((k, c, \Delta)\)-system is well approximated as

\[
E[T] \approx \frac{1}{\mu} (H_k - \frac{c}{c+1} H_{k-b_k}).
\]

Expected cost with \((C^c)\) and without \((C)\) task cancellation

\[
E[C^c] = \frac{k}{\mu} E[C] = (c(1-q) + 1) \frac{k}{\mu},
\]

where \( q = 1 - e^{-\mu \Delta} \).

**Theorem 2.** Under the shifted exponential task execution time \( X \sim S\text{Exp}(\frac{D}{k}, \mu) \), expected latency in the replication \((k, c, \Delta)\)-system is well approximated as

\[
E[T] \approx \frac{D}{\mu} + \frac{1}{\mu} (H_k - \frac{c}{c+1} H_{k-b_k}), \quad \text{where} \quad q = 1 - e^{-\mu \Delta}.
\]

Expected cost with \((C^c)\) and without \((C)\) task cancellation

\[
E[C^c] = D + \frac{k}{\mu} (1 + c(1-q - e^{-\mu \Delta})), \quad \Delta > \frac{D}{k},
\]

where \( q = 1 - e^{-\mu (\Delta - \frac{D}{k})} \).

**Expected Latency and Cost with Coding:**

**Theorem 3.** Under the exponential task execution time \( X \sim \text{Exp}(\mu) \), expected latency in coded redundancy \((k, n, \Delta)\)-system is well approximated as

\[
E[T] \approx \Delta - \frac{1}{\mu} B(q; k+1, 0) + H_{n-k} - H_{n-k}. \quad \text{where} \quad q = 1 - e^{-\mu \Delta}.
\]

Expected cost with \((C^c)\) and without \((C)\) task cancellation

\[
E[C^c] = \frac{k}{\mu} E[C] = q^k \frac{k}{\mu} + \frac{n}{\mu} (1 - q^k),
\]

where \( q = 1 - e^{-\mu \Delta} \).

**Theorem 4.** Under the shifted exponential task execution time \( X \sim S\text{Exp}(\frac{D}{k}, \mu) \), expected latency in coded redundancy \((k, n, \Delta)\)-system is well approximated as

\[
E[T] \approx \frac{D}{\mu} + \frac{1}{\mu} B(q; k+1, 0) + H_{n-k} - H_{n-k}. \quad \text{where} \quad q = 1 - e^{-\mu \Delta}.
\]

Expected cost with \((C^c)\) and without \((C)\) task cancellation

\[
E[C^c] = q^k \frac{k}{\mu} (\frac{D}{1 + \mu}) + (1 - q^k) n \left( \frac{1}{\mu} + \frac{D}{k} \right),
\]

where \( q = 1 - e^{-\mu \Delta} \).

\[
E[C^c] \approx q^k \frac{k}{\mu} \left( \frac{n-k}{n} (1-q^k) \right) + (1-q^k) n \left( \frac{1}{\mu} + \frac{D}{k} \right),
\]

\[
e^{-\eta} B(\eta; k-1, 0) \left( q^k - q^k \right).
\]

where \( q = 1(\Delta > \frac{D}{k}) (1 - e^{-\mu (\Delta - \frac{D}{k})}) \), \( q = 1 - e^{-\mu \Delta} \) and \( \eta = 1 - e^{-\mu \Delta} \).

**Scheme Comparison:** In order to answer the title question which clones to send and when, we next compare replicated and coded redundancy in distributed computing context, where it is feasible to cancel the remaining redundant tasks upon the job completion.

With exponential task execution time, under both replicated and coded redundancy, the expected cost depends neither on the time \( \Delta \) redundancy is introduced nor on the degree of redundancy \( c \) and \( n \) (see Thm 1 and 3). Consequently, in order to achieve the minimum latency, one can introduce all available redundancy at once (\( \Delta = 0 \)) with zero expected penalty in cost.

We want to understand the reduction in cost (gain) and increase in latency (pain) per increase in \( \Delta \). Fig. 2 shows cost vs. latency under delayed redundancy for \( S\text{Exp} \) tasks. For coded redundancy, we observe two phases: 1) Initially, increasing \( \Delta \) away from 0 returns almost no reduction in cost but significantly increases latency. 2) Beyond a certain point, increasing \( \Delta \) further reduces cost significantly while not increasing delay much. In other words, significant reduction in cost by delaying redundancy is possible only with significant increase in latency. Therefore, delaying coded redundancy is not effective because one can simply achieve less cost for the same latency by decreasing degree of redundancy. Simulations show that this two-phase behavior exists for Pareto task execution time as well. Note that delaying is effective for replicated redundancy to reduce cost up to some point, beyond which, once again, it is better to reduce the degree of replication.

![Figure 2: Under SExp task execution time, achievable expected cost with task cancellation vs. latency region is plotted for replicated (c = 1, 2) and coded (n \( \in \{k+1, 3k\} \)) redundancy by varying the time (\( \Delta \)) of introducing redundancy along each curve.](image)

Thm. 5 gives exact expressions for the expected cost and latency under zero-delay redundancy. Under both \( S\text{Exp} \) and Pareto task execution time, coding always achieves better expected cost and latency than replication as illustrated in Fig. 3.

**Theorem 5.** Let expected latency and cost with task cancellation be \( E[T(c,n)] \), \( E[C(c,n)] \) for zero-delay replicated redundancy, and \( E[T(k,n)] \), \( E[C(k,n)] \) for zero-delay coded re-
dundancy. Under task execution time $X \sim SExp(D/k, \mu)$,
$$E[T_{(k,c)}] = \frac{D}{k} + \frac{H_n}{(c+1)\mu}, \quad E[C_{(k,c)}] = (c+1)D + \frac{k}{\mu},$$
$$E[T_{(k,n)}] = \frac{D}{k} + \frac{1}{\mu}(H_n - H_{n-k}), \quad E[C_{(k,n)}] = \frac{nD}{k} + \frac{k}{\mu}.$$ 

Under task execution time $X \sim \text{Pareto}(\lambda, \alpha)$,
$$E[T_{(k,c)}] = \lambda k \Gamma(1 - ((c+1)\alpha)^{-1}),$$
$$E[C_{(k,c)}] = \lambda k \frac{(c+1)^{\alpha}}{(c+1)^\alpha - 1},$$
$$E[T_{(k,n)}] = \frac{\lambda n}{(n-k)!} \left[ \Gamma(n-k+1-\alpha^{-1}) - \Gamma(n-1-\alpha^{-1}) \right],$$
$$E[C_{(k,n)}] = \frac{\lambda n}{\alpha - 1} \left[ \Gamma(n) - \Gamma(n-k+1-\alpha^{-1}) \right] \Gamma(n-k) \Gamma(n+1-\alpha^{-1}).$$

One would expect that adding more redundancy reduces latency but always increases cost. In [4] replicated redundancy is demonstrated to reduce both cost and latency under heavy-tail task execution time. Fig. 3 shows and compares this for replicated and also the coded redundancy using the analytical expressions presented here. Under heavy-tail, it is possible to reduce latency by adding redundancy and still pay for the baseline cost of running without redundancy. Corollary 1 gives expressions for the minimum achievable expected latency without exceeding the baseline cost.

**Corollary 1.** Under task execution time $X \sim \text{Pareto}(\lambda, \alpha)$ in zero-delay replicated redundancy system, minimum latency $E[T_{\text{min}}]$ that can be achieved without exceeding the baseline cost is

$$E[T_{\text{min}}] = \lambda k \Gamma(1 - (\alpha(c_{\text{max}}+1))^{-1}) \Gamma(k+1 - (\alpha(c_{\text{max}}+1))^{-1}).$$

where $c_{\text{max}} = \max\left\{ \left(\frac{1}{\alpha-1}\right) - 1, 0 \right\}$ and any reduction in latency without exceeding the baseline cost is possible only if $\alpha < 1.5$. For coded redundancy system, a tight upper bound on $E[T_{\text{min}}]$ is

$$E[T_{\text{min}}] < \alpha + \lambda k \Gamma(1 - \alpha^{-1}) \Gamma(k+1 - \alpha^{-1}).$$

Fig. 4 illustrates that the maximum percentage reduction in latency while paying for less than the baseline cost depends on the tail of task execution time. As stated in Corollary 1, this reduction is possible under replicated redundancy only when the tail index is less than 1.5, in other words when the tail is very heavy, while coding relaxes this constraint significantly. In addition, the constraint on $\alpha$ is independent of the number of tasks $k$ under replication, while it increases with $k$ under coding, meaning that jobs with larger number of tasks can get reduction in latency at no cost even for lighter tail task execution times.

**3. REFERENCES**


Networked Cournot Competition in Platform Markets: Access Control and Efficiency Loss

Weixuan Lin†, John Z. F. Pang†, Eilyan Bitar‡, Adam Wierman†
† California Institute of Technology, Pasadena, CA, USA, Email: {jpang, adamw}@caltech.edu
‡ Cornell University, Ithaca, NY 14853, USA, Email: {wl476,eyb5}@cornell.edu

ABSTRACT

This paper studies network design and efficiency loss in open and discriminatory access platforms under networked Cournot competition. In open platforms, every firm connects to every market, while discriminatory platforms limit connections between firms and markets to improve social welfare. We provide tight bounds on the efficiency loss of both platforms: (i) that the efficiency loss at a Nash equilibrium under open access is bounded by 3/2, and (ii) for discriminatory access platforms, we provide a greedy algorithm for optimizing network connections that guarantees efficiency loss at a Nash equilibrium is bounded by 4/3, under an assumption on the linearity of cost functions.

1. INTRODUCTION

Platforms have changed the way entire industries are run, e.g., ride-sharing. Unlike traditional firms, platforms do not manufacture products or provide a service. Instead, they arrange matches between firms and consumers, and facilitate a safe and simple trading process, providing value for all parties involved. Today, platforms, e.g., Facebook, Uber, Amazon, Ebay, make up a $3 trillion market [1].

The design and operation of platforms is extremely diverse, e.g. platforms like Amazon aim to match buyers to sellers taking into account sellers’ prices and reviews and buyers’ preferences while others use pricing and allocation. There are two schools of thought in platform design: (i) Open access, where the platform allows firms and markets to make their own choices on matching and allocations, or (ii) Discriminatory access, where the platform restricts the set of markets each firm is allowed to enter to promote economical efficiency [8]. Examples of open access platforms include eBay, and examples of discriminatory access platforms include Amazon’s Buy Box. Open access and discriminatory access designs are approaches with differing benefits. Open access designs are simpler to maintain, completely transparent, and provide fairness across firms. On the other hand, discriminatory access offers the platform additional control to optimize social welfare, at the expense of complexity, transparency, and fairness.

Thus, the question is, how large an improvement in efficiency is possible by moving from open access to discriminatory access?

1.1 Contributions of this paper

We provide tight efficiency results for both open access and discriminatory access platform designs; quantifying the improvements in efficiency that discriminatory access designs can provide. Concretely, this paper builds on recent work [15], that studies platform design using the model of networked Cournot competition. In the context of this model, this paper makes two main contributions.

1. Open access efficiency loss bound of $\frac{3}{2} \left(1 - \frac{1}{3n} + \frac{1}{2n^2}\right)$.
2. Greedy algorithm for optimal discriminatory network under linear cost functions, with a 4/3 efficiency loss bound.

1.2 Related work

Our work relates to both platform design and networked competition:

Platform design: Recent growth of online platforms has led researchers to focus on identifying design features common to successful platforms. Work in this area has covered a variety of design factors, including pricing [16] and competition [4]. Empirical findings display significant price dispersion in online marketplaces [10], causing platforms to differentiate products in order to create distinct consumer markets [9]. In particular, these results highlight the need to study platforms in the context of networked competition.

Competition in networked settings: Models of networked competition aim to capture the effect of network constraints on the strategic interaction between firms. These models include networked Bertrand competition, e.g., [7], networked Cournot competition, e.g., [2, 5], and other non-cooperative bargaining games where agents trade via bilateral contracts and a network determines the set of feasible trades, e.g., [3, 14].

Our work fits squarely into the setting of networked Cournot competition. A large swath of literature on networked Cournot competitions, e.g., [2, 5], focuses on characterizing and computing Nash equilibria. Recent streams of literature that closely relate to our work are (i) characterizing the efficiency loss of networked Cournot games [11, 15] and (ii) understanding the impacts of system operator governance on the resulting Nash equilibria [6]. This paper is the first to provide a tight bound on the efficiency loss of open access platforms, improving on the bounds in [15], and the first to provide an algorithm for network design with provable guarantees.

2. MODEL AND PRELIMINARIES

We describe competition in online platforms according to the networked Cournot competition model first introduced by [2] and [5], and later employed by [15] to describe competition in platforms.

Network and Platform Models

The network specifying connections between firms and markets is described by a bipartite graph $(F, M, \delta)$, where $F := \{1, \ldots, n\}$ is the set of $n$ firms and $M := \{1, \ldots, m\}$ the set of $m$ markets, with $\delta \subseteq F \times M$ the set of directed edges connecting firms to markets. Open access platforms, allowing all firms to access all markets, corresponds to a complete bipartite graph. Discriminatory access platforms restrict the set of markets accessible by each firm, with the goal to improve social welfare.
Producer Model

Under both platform models, each firm $i$ specifies the quantity $q_i \in \mathbb{R}_+$ it produces in each market $j$, and $q_{ij} \in \mathbb{R}^n_{\geq 0}$ denotes the supply profile from firm $i$. We require that $q_{ij} = 0$ for all $(i, j) \notin E$, and define the set of feasible supply profiles from firm $i$ as $\mathcal{Q}_i(E) = \{x \in \mathbb{R}^m_n \mid x_j = 0, \forall (i, j) \notin E\}$. We denote the supply profile from all firms by $q = (q_1, \ldots, q_n) \in \mathbb{R}^m_n$ and the set of profiles from all firms by $\mathcal{Q}(E) = \prod_{i=1}^n \mathcal{Q}_i(E)$.

The production cost of each firm $i \in F$ depends on its supply profile only through its aggregate production quantity, which is given by $s_i := \sum_{j \in F} q_{ij}$. The production cost of firm $i$ is defined by $C_i(s_i)$, where we assume that the cost function $C_i$ is convex, differentiable on $(0, \infty)$ and satisfies $C_i(s_i) = 0$ for all $s_i \leq 0$. Finally, we define $C := (C_1, \ldots, C_n)$ as the cost function profile.

Market Model

As is standard in Cournot models, we model price formation according to an inverse demand function in each market. Following [5], we focus on affine inverse demand functions throughout this paper. Specifically, the price in each market $j \in M$ is determined according to $p_j(d_j) := a_j - b_j d_j$, where $d_j$ denotes the aggregate quantity being produced in market $j$. It is given by $d_j := \sum_{i \in F} q_{ij}$. Here, $a_j > 0$ measures consumers’ maximum willingness to pay, and $b_j > 0$ the price elasticity of demand.

The Networked Cournot Game

We describe the equilibrium of the market specified above according to Nash. In particular, we consider profit maximizing firms, where the profit of a firm $i$, given the supply profiles of all other firms $q_{-i} = (q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n)$, is given by

$$\pi_i(q_i, q_{-i}) := \sum_{j=1}^m q_{ij} p_j(d_j) - C_i(s_i),$$

where $\pi := (\pi_1, \ldots, \pi_n)$ is the set of payoff functions of all firms.

The triple $(F, \mathcal{Q}(E), \pi)$ defines a normal-form game, which we refer to as the networked Cournot game associated with the edge set $E$. Its Nash equilibrium is defined as follows:

**Definition 1.** A supply profile $q \in \mathcal{Q}(E)$ constitutes a pure strategy Nash equilibrium of the game $(F, \mathcal{Q}(E), \pi)$ if for every firm $i \in F$, $\pi_i(q_i, q_{-i}) \geq \pi_i(q'_i, q_{-i})$, for all $q'_i \in \mathcal{Q}_i(E)$.

Under the assumptions of convex cost functions and affine inverse demand functions, [2] has shown networked Cournot games associated with any edge set admit unique Nash equilibria.

Social Welfare and the Price of Anarchy

In this paper, we measure the performance (or efficiency) of a platform according to social welfare. For platforms, the pursuit of social welfare benefits both buyers and sellers, and in the long run, promotes their expansion.

The social welfare associated with a supply profile $q$ and a cost function profile $C$ is defined according to

$$SW(q, C) := \sum_{j=1}^m \int_0^{d_j} p_j(z)dz - \sum_{i=1}^n C_i(s_i).$$

Further, we define the efficient social welfare associated with an edge set $E$ and a cost function profile $C$ as

$$SW^*(E, C) := \sup_{q \in \mathcal{Q}(E)} SW(q, C).$$

and any supply profile $q \in \mathcal{Q}(E)$ that attains it is said to be efficient.

In general, the Nash equilibrium of the networked Cournot game will deviate from the efficient supply profile. We measure this loss of efficiency according to the price of anarchy of the game [12].

**Definition 2.** The price of anarchy associated with the edge set $E$, the cost function profile $C$, and the corresponding networked Cournot game $(F, \mathcal{Q}(E), \pi)$ is defined as

$$\rho(E, C) := \frac{SW^*(E, C)}{SW(q^{NE}(E), C)}.$$  

We set $\rho(E, C) = 1$ if $SW^*(E, C) / SW(q^{NE}(E), C) = 0/0$.

3. OPEN ACCESS PLATFORMS

For our first set of results, we focus on providing tight bounds on the price of anarchy of the networked Cournot game in an open access platform, that depends both on the number of firms and the degree of asymmetry between firms’ cost functions. These results improve upon the bounds in [15] and generalize those in [11].

3.1 Identifying the Worst-case Cost Function Profile

The following lemma establishes piecewise linearity as the worst-case cost function profile.

**Lemma 1.** Given a cost function profile $C$, define the cost function profile $C = (C_1, \ldots, C_n)$ according to

$$C_i(s_i) = \left(C'_i \left(\sum_{j=1}^m d_{ij} \langle x_{FMj} \rangle \right) s_i \right)^+$$

for $i = 1, \ldots, n$. It holds that $\rho(\langle x_{FM}, C \rangle) \leq \rho(\langle x_{FM}, C \rangle) \leq \rho(\langle x_{FM}, C \rangle) = 0/0$.

In words, given any cost function profile $C$, it is always possible to construct another cost function profile $C$ consisting of (piecewise) linear functions, which has a price of anarchy that is no smaller. Therefore, in constructing a price of anarchy bound that is guaranteed to hold for all cost functions belonging to the family specified in Section 2, it suffices to consider cost functions that are linear on $(0, \infty)$.

3.2 Efficiency Loss in Open Access Platforms

We examine the role played by (asymmetry) in the cost function profile in determining platform efficiency.

3.2.1 Symmetric Cost Functions

We begin with the setting in which firms have identical cost functions and propose the following:

**Proposition 1.** If $C_1 = C_2 = \ldots = C_n$, then the price of anarchy associated with the corresponding open access networked Cournot game $(F, \mathcal{Q}(E), \pi)$ is bounded (tight) by

$$\rho(\langle x_{FM}, C \rangle) \leq 1 + \frac{1}{(n+1)^2 - 1}.$$  

This conforms with the intuition that increasing the number of (symmetric) suppliers increases competition, and thereby reduce the extent to which any one producer might exert market power.

---

2 Implicit in our definition of the price of anarchy for the networked Cournot game is the fact that the networked Cournot game admits a unique Nash equilibrium. In general, for games with a possible multiplicity of Nash equilibria, the price of anarchy is defined as the ratio of the efficient social welfare over that of the Nash equilibrium with the worst social welfare.
3.2.2 Arbitrary Asymmetric Cost Functions

We now consider the general setting where firms have arbitrarily asymmetric cost.

**Theorem 1.** The price of anarchy associated with a cost function profile $C$ and the corresponding open access networked Cournot game $(F, 2\{X_{FM}\}, \pi)$ is upper bounded (tightly) by

$$\rho(X_{FM}, C) \leq \frac{3}{2} \left( 1 - \frac{1}{3n+6} \right).$$

Taking the number of firms $n \to \infty$ yields a price of anarchy bound that is valid for any number of firms or markets. This recovers the 3/2 price of anarchy bound first established in [11] for a single market, and improves upon the previously known 16/7 price of anarchy bound for open access in [15].

4. DISCRIMINATORY ACCESS PLATFORMS

Here, the platform specifies the edge set of the bipartite graph that connect firms to markets to maximize the social welfare at the unique Nash equilibrium of the resulting networked Cournot game.

We state without proof that finding the optimal edge set maximizing social welfare at Nash equilibrium amounts to a mathematical program with equilibrium constraints (MPEC), and is, in general, intractable. Under the assumption that each firm’s cost function is linear, we present a greedy algorithm that obtains an optimal network design. Furthermore, we present a tight price of anarchy bound for the resulting game under discriminatory access.

4.1 A Greedy Algorithm for Linear Cost Functions

We propose the following greedy algorithm for solving the optimal network design problem. For each market $j \in M$, the algorithm visits firms in ascending order of marginal cost, and provides each firm access to market $j$ if its inclusion improves social welfare.

**Algorithm 1** The Greedy Algorithm

**Require:** $c_1 \leq \cdots \leq c_n$.

1. Initialize edge set $E' \leftarrow \emptyset$.
2. for $j = 1$ to $n$ do
3. Initialize firm index $i \leftarrow 1$.
4. Initialize edge set $E' \leftarrow E'$.
5. repeat
6. Update edge set $E' \leftarrow \tilde{E}$.
7. if $i \leq e$ then
8. Set edge set $E' \leftarrow E' \cup (i, j)$.
9. Set firm index $i \leftarrow i + 1$.
10. end if
11. until $SW(q^{NE}(\tilde{E}), C) \leq SW(q^{NE}(E'), C)$.
12. end for
13. return $E'$.

The implementation of Algorithm 1 yields an edge set $E^*$, whose corresponding Nash equilibrium has social welfare no smaller than that of the open access one. In the following theorem, we quantify this improvement in social welfare via a tight bound on the price of anarchy in discriminatory access networked Cournot games.

**Theorem 2.** Assume that each firm’s cost function is linear. The price of anarchy of discriminatory access platforms is $4/3$.

5. CONCLUDING REMARKS

This paper examines the design and efficiency loss of open and discriminatory access platforms. Open access platforms provide transparency, while discriminatory access platforms provide additional control leveraged to improve efficiency. For open access platforms, we establish a tight upper bound on the price of anarchy (PoA) that is decreasing (increasing) in the number of firms, when costs are symmetric (asymmetric). Consequently, we show that open access platforms have a PoA at most $3/2$.

We contrast this bound with the case of discriminatory access platforms. Assuming that firms’ costs are linear, we propose and prove the optimality of a greedy algorithm, recovering the optimal network design for discriminatory access platforms. In this setting, we show that the PoA bound shrinks to $4/3$, thereby improving upon the worst-case efficiency loss of open access platforms.

Our work builds on a growing literature studying networked Cournot competition, including [2, 5, 6, 15]. While this literature is maturing, there are still a wide variety of important open questions that remain. For example, the problem of constructing approximation algorithms with provable bounds on performance arises as an interesting direction for future work.

A full version of this work (including proofs) is found in [13].

6. REFERENCES

[1] Taming the beasts; regulating technology companies. The Economist (US), May 2016.


When is Anarchy Beneficial?

Tyler Maxey
University of California
Berkeley, CA 94720
tylermaxey@berkeley.edu

Hakjin Chung
KAIST School of Business
85 Hoegiro, Dongdaemun-gu
Seoul, Korea, 02455
hakjin@business.kaist.ac.kr

Hyun-Soo Ahn
Ross School of Business
University of Michigan
Ann Arbor, MI 48109
hsahn@umich.edu

Rhonda Righter
University of California
Berkeley, CA 94720
rrighter@ieor.berkeley.edu

ABSTRACT
In many service systems, customers acting to maximize their individual utility (selfish customers) will result in a policy that does not maximize their overall utility; this effect is known as the Price of Anarchy (PoA). More specifically, the PoA, defined to be the ratio of selfish utility (the overall average utility for selfish customers) to collective utility (the overall average utility if customers act to maximize their overall average utility) is generally less than one. Of course, when the environment is fixed, the best case PoA is one, by definition of the maximization problem. However, we show that in systems with feedback, where the environment may change depending on customer behavior, there can be a Benefit of Anarchy, i.e., we can have a PoA that is strictly larger than one. We give an example based on a Stackelberg game between a service provider and customers in a single-server queue.

Keywords
Stochastic model applications; Probability; Markovian queues

1. INTRODUCTION
In this paper, we examine the effects of selfish behavior and explore the question of whether selfishness is ever beneficial. In particular, we consider a scenario where arriving customers decide whether or not to join a single-server queue, to maximize either their individual or their joint (collective) utility. We explore this effect through the Price of Anarchy (PoA), which is the ratio of individual customer utility (their utility when they are acting selfishly) to collective customer utility (their utility when they are acting to maximize overall average utility). It follows by definition that PoA is at most one for a fixed environment. We consider a model in which, knowing the behavior (selfishness) of the arrivals, the firm sets the service rate to maximize its profit, and show that individual customer utility can be higher than collective customer utility in this case.

There has been a lot of research that examines strategic behavior in queueing systems, dating back to the seminal paper of Naor [5]. See [3, 4] for overviews. We start with Naor’s M/M/1 queueing model, with Poisson arrivals at rate λ and service rate μ, and where customers earn a reward R upon service completion, pay a price p for service, and pay a holding cost h per unit time they spend in the system. Without loss of generality we scale time so that λ = 1. In Naor’s model, customers observe the queue length. We will consider both the cases of observable and unobservable queues. The latter was studied by Edelson and Hildebrand [2]. In both cases, selfish customers do not consider the effect of their joining on later arriving customers (their negative externalities), and they join more than collective customers who are maximizing overall customer welfare. In [2, 5] the authors considered adjusting the price, p, to induce collectively optimal behavior among selfish customers; then PoA = 1 can be obtained. PoA = 1 can also be induced by changing the order of service to preemptive last-come first-served [3, 4].

We show that there can be a Benefit of Anarchy (BoA), i.e., PoA > 1, in a Stackelberg game where the firm sets the capacity to maximize its profits. Here we hold p fixed, and assume the server pays cost cμ per unit time if it chooses capacity μ. (We can generalize this to increasing convex c(μ), but we keep it simple here.)

2. RESULTS: UNOBSERVABLE QUEUE
For the unobservable queue, for fixed μ, customers will join the queue with probability q_μ(μ), where α ∈ {S, C} for selfish and collective customers, respectively, so the effective arrival rate is q_μ(μ)λ = q_μ(μ) because we set λ = 1. The overall customer utility is

\[ U_α(μ) := q_α(μ) \left( R - p - \frac{1}{μ - q_α(μ)} \right) \]

where \( \frac{1}{μ - q_α(μ)} \) is the average waiting time (which is also the waiting cost, because we set h = 1) of joining customers, and where we assume R > p. (Otherwise, q_μ(μ) = 0 for all μ.) We will see that the joining decision, for α ∈ {S, C}, will be defined by μ^α_0 and μ^α*_0 such that q_μ(μ) = 0 for μ ≤ μ^α_0, q_μ(μ) = 1 for μ ≥ μ^α*_0, and 0 < q_μ(μ) < 1 for μ^α_0 < μ < μ^α*_0.

The firm chooses μ to maximize its profit, \( P_α(μ) := p q_α(μ) - cμ \), where we assume p > c; otherwise the firm chooses \( μ^α_0 = 0 \). We will see that the optimal choice for the firm is \( μ^α_0 ∈ \{0, μ^α*_0\} \), depending on whether \( q_α(μ) \) is smaller or larger than \( μ^α_0 \). The Price of Anarchy for fixed μ is defined as...
PoA(µ) := U_S(µ)/U_C(µ), and for the feedback system it is PoA := U_S(µ^*)/U_C(µ^C). For convenience, we define \( b := \frac{1}{R} \).

Given µ, we observe three cases of the Nash Equilibrium for selfish customers:

(i) If \( R < \frac{1}{b} \) then \( q_S(\mu) = 0 \), which clearly makes \( U_S(\mu) = 0 \), and the best value of \( \mu \) for the firm in this range is \( \mu = 0 \), with \( \Pi_S(0) = 0 \).

(ii) If \( R \geq \frac{1}{b} \) then \( q_S(\mu) = 1 \). Because \( \Pi_S(\mu) \) is decreasing in this range, the best \( \mu \) is \( \mu = \mu_S^1 = b + 1 \), with \( \Pi_S(\mu_S^1) = p - c(b + 1) \) and \( U_S(\mu) = 0 \).

(iii) If \( \mu \in (\mu_S^1, \mu_S^2) \) then customers employ a mixed strategy by joining with probability \( q_S(\mu) \) such that
\[
R - p = \frac{1}{\mu_S^2 - \mu_S^1} \Rightarrow q_S(\mu) = \frac{\mu - b}{\mu_S^2 - \mu_S^1}.
\]

The profit for selfish customers when the threshold is \( n \) is
\[
\Pi_S(\mu) = p(\mu - b) - \mu c, \quad \text{where is increasing in } \mu, \text{ so the best } \mu \text{ is } \mu = \mu_S^1 = b + 1, \text{ with profit } \Pi_S(\mu_S^1) = p - c(b + 1) \text{ and } U_S(\mu) = 0.\]

Summarizing, selfish customers, the firm chooses \( \mu_S^* = \mu_S^1 I\{ \frac{\mu}{\mu} > \mu_S^1 \} \), where \( I\{ \cdot \} \) is the indicator function.

Because the utility of the selfish customers is 0 in the feedback system, the Price of Anarchy is PoA = 0. However, it is interesting to see how the utilities for selfish and collective customers compare, so we define a Cost of Anarchy,  
\[
CoA(\mu) = U_C(\mu) - U_S(\mu) \text{ and } CoA = U_C(\mu^C) - U_S(\mu^S) \geq 0.
\]

For a fixed \( \mu \), collective customers choose \( q_C(\mu) \) to maximize their overall utility, \( U_C(\mu) \). We observe three cases:

(i) If \( \mu \leq b := \mu_C^1 = \mu_C^0 \), then \( q_C(\mu) = 0 \), \( U_C(\mu) = 0 \), and the best \( \mu \) is \( \mu = 0 \).

(ii) If \( \mu > \mu_C^2 \) then customers always join, i.e., \( q_C(\mu) = 1 \). In this case, the firm’s profit decreases with \( \mu \), so the best \( \mu \) is \( \mu_C^1 \), with \( \Pi_C(\mu_C^1) = p - \mu C^1 = p - \frac{1}{2}(2 + b + \sqrt{b^2 + 4b}) \) and \( U_C(\mu_C^1) = R - p - \frac{1}{\mu_C^1} = \frac{1}{2} - \frac{b + \sqrt{b^2 + 4b}}{\mu_C^1} \).

(iii) If \( 0 < \mu < \mu_C^0 \), collective customers join with probability \( q_C(\mu) = p - \mu C^1 = p - \frac{1}{2}(2 + b + \sqrt{b^2 + 4b}) \) and \( U_C(\mu_C^1) = R - p - \frac{1}{\mu_C^1} = \frac{1}{2} - \frac{b + \sqrt{b^2 + 4b}}{\mu_C^1} \).

Summarizing, collective customers, the firm chooses \( \mu_C^* = \mu_C^1 I\{ \frac{\mu}{\mu} > \mu_C^1 \} \), where \( I\{ \cdot \} \) is the indicator function.

We have that \( q_S(\mu) \geq q_C(\mu) \), so collective customers join less often for a fixed \( \mu \). We also have \( \mu_C^1 > \mu_S^1 \), but we may have \( \mu_C^1 < \mu_S^1 \) if \( \mu_S^1 < \mu_C^1 \). Proposition 1 tells us that having a strategic firm increases the Cost of Anarchy if the firm chooses to serve collective customers, and otherwise decreases it (to 0). See Figure 1 for an example illustrating the result when it is profitable for the firm to serve both types of customers.

**Proposition 1.** If \( \mu_C^0 > 0 \), then \( \mu_C^1 > \mu_S^1 > 0 \) and \( CoA = CoA(\mu) \geq 0 \); otherwise \( CoA = CoA(\mu) \geq 0 \).

**Proof.** The result follows by observing that if \( \mu_C^0 > 0 \) then \( \mu_C^1 = \mu_C^0, CoA(\mu) = 0 \) for \( \mu = \mu_C^0, CoA(\mu) \leq UC(\mu), CoA = UC(\mu_C^1) \) and utility is increasing in \( \mu \).

We now consider how the parameters change the strategic service rate and the CoA. It can be shown, for \( \alpha = \{ S, C \}, R > R_\alpha \), and \( a_\alpha < p < b_\alpha \), that \( \mu^*_\alpha \) is decreasing convex in \( R \) and increasing convex in \( p \), and CoA is increasing convex in \( R \) and decreasing convex in \( p \).

![Figure 1: Cost of Anarchy (R = 10, p = 5, c = 3).](image1)

**3. RESULTS: OBSERVABLE QUEUE**

We know from [5] that there exist thresholds \( n_C(\mu) \leq n_S(\mu) \) such that a selfish (collective) customer will join the queue if the number of customers upon arrival is less than or equal to
\[
n_S(\mu) = \max \{ n \in N_0 \mid R - p - \frac{h(n - 1)}{\mu} \geq 0 \} = \lfloor b \mu \rfloor
\]
\[
n_C(\mu) = \max \{ n \in N_0 \mid p n^{n+1} - (n + 1) n^{n+1} + 1 \leq b \mu n^{n-1} \}.
\]

where \( \lfloor \cdot \rfloor \) is the largest integer less than or equal to the value in the bracket.

The firm chooses \( \mu \) to maximize its profit, \( \Pi_n(\mu, n(\mu)) = p \lambda(\mu, n(\mu)) - \mu c \mu \), where \( \lambda(\mu, n) \) is the effective arrival rate when the threshold for joining is \( n \) and the service rate is \( \mu \). Because the threshold is a step function in \( \mu \), the profit function is discontinuous in \( \mu \), and the problem of choosing \( \mu \) to maximize profit is much more difficult than in the unobservable case. We were able to show the properties below, which reduces the difficulty of finding an optimal solution [1]. Refer to Figure 2.

![Figure 2: The original profit and the relaxed profit when R = 10, p = 5, c = 3.](image2)
threshold for entering will be n; we call this the threshold interval.

**Proposition 2.** For both selfish and collective customers,

(i) $\Pi_\alpha(\mu, n_\alpha(\mu))$ is strictly concave in $\mu$ for $\mu$ within each threshold interval.

(ii) $\Pi_\alpha(\mu, n_\alpha(\mu))$ is discontinuous and jumps upward at $\mu_\alpha(n)$ for $n \in \mathbb{N}_0$.

From the proposition we know that the optimal $\mu$ can be at the left, right, or in the interior of any of the threshold intervals, so an exhaustive search through each interval will yield the optimal solution. (See Figure 2 for an example for selfish customers.) To further simplify the search, we consider an upper envelope for the profit function. By relaxing the integer constraint on the thresholds and defining relaxed thresholds $\tilde{n}_S(\mu) = b\mu$ and $\tilde{n}_C(\mu) = \{n \mid n\mu^{n+1}-(n+1)\mu^n+1 = b\}$, the relaxed profit function,

$$\tilde{\Pi}_\alpha(\mu, \tilde{n}_\alpha(\mu)) = p\left(1 - \frac{\mu - 1}{\mu^{n_\alpha(\mu)+1} - 1}\right) - c\mu,$$

is more amenable to analysis and allows us to more efficiently determine the optimal service rate.

**Proposition 3.** The following properties hold for both customer types:

(i) $\tilde{\Pi}_\alpha(\mu, \tilde{n}_\alpha(\mu))$ is differentiable, is an upper envelope of the profit function, and is equal to the actual profit if and only if $\mu = \mu_\alpha(n)$ for all $n$.

(ii) The optimal service rate exists in an interval that contains a maximizer of the relaxed profit function.

With the relaxed profit function, we can specify intervals that may contain the profit-maximizing service rate through first order analysis, so we can restrict our search to intervals that contain stationary points of the envelope function. However, the analysis is still difficult because, in the collective case, the threshold does not exist in a closed form, and the relaxed profit function is not necessarily concave.

**Figure 3: PoA $\geq 1$ when $R = 10, p = 5, c = 3$.**

We saw that for the unobservable queue, the firm will choose a higher service rate when customers are collective than when they are selfish, and the utility for selfish customers is 0. For the observable queue, selfish customers can have positive utility, and they may also induce a higher service rate from the firm. Thus, we have the possibility of a Benefit of Anarchy! Figure 3 illustrates a case when PoA$>1.18$, so selfish customers have an average utility that is 18% higher than that of collective customers. (With nonlinear costs for server speed, we have obtained examples with PoA$>1.8$.) From the figure it is clear that although $U_C(\mu) \geq U_S(\mu)$ for all fixed $\mu$, we have $\mu^*_C < \mu^*_S$ and $U_C(\mu^*_C) < U_S(\mu^*_S)$.

The notion of a Benefit of Anarchy is intriguing, but it is difficult to pinpoint. While we know that BoA requires $\mu^*_C < \mu^*_S$, this is not a sufficient condition. Because of the difficulties in finding $\mu^*_C$ noted above, it is not possible to give an explicit region of the parameter space where we know BoA occurs. After plotting the PoA across a range of two parameters, holding all else constant (refer to Figure 4), it is easy to see why there are not simple, explicitly characterizable regions where there is a Benefit of Anarchy (PoA$>1$). In Figure 4 we choose to vary $h$ rather than $R$ or $p$, in addition to $c$, because this effect shows up more clearly in the graph. (Essentially, we are varying $\frac{1}{\pi^2p} = b$.)

We note that because $n_S(\mu) \geq n_C(\mu)$, $\Pi_S(\mu, n_S(\mu)) \geq \Pi_C(\mu, n_C(\mu))$, and thus $\Pi_S := \Pi_S(n_S^*, n_S(\mu^*_S)) \geq \Pi_S(n_C(\mu^*_S)) \geq \Pi_C := \Pi_C(\mu^*_C, n_C(\mu^*_C))$. That is, the firm always benefits from anarchy. Therefore, when there is a Benefit of Anarchy, both the customers and the firm benefit, and overall social welfare is higher. We have also shown in [1] that there can be a Benefit of Anarchy when the server is operated by a social, or regulated, firm, or government agency, that chooses the service rate to maximize social welfare.

**Figure 4: PoA when $R = 10, p = 5$.**

4. REFERENCES


1. INTRODUCTION

Caches play a prominent role in networks and distributed systems, and their importance is reflected in much recent work on performance analysis of caching algorithms. A plethora of research has been done on the analysis of caching algorithms using the metric of hit probability under the Independent Reference Model (IRM). However, there has been a tremendous increase in the demand of different types of content with different quality of service requirements; consequently, the user needs in the networks become more heterogeneous. In order to meet such challenges, the design and analysis of content delivery networks need to incorporate service differentiation among different classes of contents and applications. Though considerable literature has focused on the design of fair and efficient caching algorithms for content distribution, little work has focused on the provision of multiple levels of service in network and web caches.

A related problem has been considered in [1], where the authors formulated a Hit-probability Based Cache Utility Maximization (HPB-CUM) framework under the IRM and constant content size

$$\text{HPB-CUM} : \max \sum_{i=1}^{n} U_i(h_i^p), \quad \text{s.t.} \quad \sum_{i=1}^{n} h_i^p = B, \quad (1)$$

where $n$ is the number of unique contents in the system, $B$ is the cache size, $h_i^p$ is the stationary hit probability of content $i$, and $U_i : [0,1] \rightarrow \mathbb{R}$ is the utility function. The paper characterized the optimal TTL cache policies [3] with an increasing, continuously differentiable, and strictly concave utility function, and also proposed an online algorithm for cache management.

While the characterization of cache management with respect to (w.r.t.) hit probability is valuable, hit rate [2] is a more generic performance metric given the request arrival rate in real systems. For example, pricing based on request rate is preferable to that based on cache occupancy by a service provider. Furthermore, the goal of a service provider in designing hierarchical caches might be to minimize the internal bandwidth cost, which can be characterized with a utility function $U_i = -C_i(m_i)$, where $C_i(m_i)$ is the cost associated with miss rate $m_i$ for content $i$. Hence, it is insufficient to only identify the optimal cache management w.r.t.

hit probability, and one also needs to characterize the optimal policy when utility is measured as a function of hit rate.

Therefore, it is reasonable to define utility functions as functions of hit rates. Then a fundamental research question is: when hit rate based utility maximization is favorable than hit probability based utility maximization? One argument in support of utility maximization as functions of hit rate is that it is more natural from the perspective of content providers. Since the utility function should not only capture the impact of hit probability, but also the amount of data arriving at the system, which is characterized by the hit rate. However, performance analysis results on hit rate with utility maximization are relatively few. The typical performance analysis under the IRM usually follows a fixed Zipf distribution that models a heavy tail popularity distribution observed in empirical studies. One objective in this paper is to formulate a Hit-rate Based Cache Utility Maximization (HRB-CUM) framework for maximizing aggregate content utility subject to buffer size constraints at the service provider. We wish to explore the tradeoff between HRB-CUM and HPB-CUM: (i) adaptability to the heavy-tailed request process, and (ii) the robustness and stability of the corresponding online-algorithms.

Towards this goal, we first formulate HRB-CUM and then derive explicit expressions for the corresponding optimal hit rate and hit probability under the family of $\beta$-fair utility functions. We compare the relative metrics under different content weights. We find that under IRM requests, there exists a threshold on the identity of content. To be more specific, if we assume that the popularity is in a non-increasing order, i.e., $p_1 \geq \cdots \geq p_n$, then for $\beta < 1$, there exists a threshold $1 < j < n$ such that HRB-CUM will favor most popular contents than HPB-CUM, i.e., popular contents will be cached under HRB-CUM. Similar arguments for $\beta > 1$, and these will be described in details in Section 3.

Although the above comparisons provide insights on the advantage of HRB-CUM over HPB-CUM in the notion of fairness, they say nothing about how well one approach can respond to the changes in the system. The relevant methodology is to propose online algorithms that can adapt to changes in the system with limited information. We show that the corresponding online algorithms for HRB-CUM is more robust and stable than HPB-CUM w.r.t. the convergence rate, details are given in Section 4.

2. MATHEMATICAL MODELS

We consider the IRM model with a Poisson arrival pro-
cess for most of our analysis. Suppose there are a set of \( n \) contents and a cache of size \( B \). Let \( \lambda_i' = \lambda_i h_i' \) denote the hit rate for content \( i \) with arrival rate \( \lambda_i \) and hit probability \( h_i' \).

**TTL Caches**: Under the TTL cache policy, content \( i \) is inserted into the cache with a timer \( t_i \) at the time of a cache miss. In particular, we consider the reset TTL cache, i.e., the TTL is reset to \( t_i \) each time content \( i \) is requested. From previous work [2], the hit rate of content \( i \) satisfies \( \lambda_i' = \lambda_i (1 - e^{-\lambda_i t_i}) \).

**Utility Function and Fairness**: Different utility functions define different fairness properties. Here, we focus on the widely used \( \beta \)-fair utility functions, defined as follows

\[
U_i(x) = \begin{cases} 
    w_i^{1 - \beta} & \beta \geq 0, \beta \neq 1; \\
    w_i \log x & \beta = 1,
\end{cases}
\]  

(2)

where \( w_i > 0 \) denotes the weight for content \( i \).

**Cache Usage Maximization**: We formulate cache management as a utility maximization problem. We introduce HRB-CUM in this section and compare it to HPB-CUM given in (1) [1]. More specifically,

\[
\text{HRB-CUM} : \quad \max \sum_{i=1}^{n} U_i(\lambda_i') \quad \text{s.t.} \quad \sum_{i=1}^{n} \lambda_i' / \lambda_i = B. 
\]  

(3)

With the Lagrangian method, we easily obtain the optimal hit rate \( \lambda_i' \) and hit probability \( h_i' \) under HRB-CUM for \( \beta \geq 0 \) and \( \beta \neq 1 \) given as follows

\[
\lambda_i' = \frac{w_i^{1/\beta} \lambda_i^{1/\beta}}{\sum_j w_j^{1/\beta} \lambda_j^{1/\beta}} B, \quad h_i' = \frac{w_i^{1/\beta} \lambda_i^{1/\beta - 1}}{\sum_j w_j^{1/\beta} \lambda_j^{1/\beta - 1}} B. 
\]  

(4)

We skip the derivations due to space constraints. From [1], the corresponding optimal hit rate and hit probability under HPB-CUM are \( \lambda_i^p = \frac{w_i^{1/\beta} \lambda_i^{1/\beta}}{\sum_j w_j^{1/\beta} \lambda_j^{1/\beta}} B \) and \( h_i^p = \frac{w_i^{1/\beta} \lambda_i^{1/\beta - 1}}{\sum_j w_j^{1/\beta} \lambda_j^{1/\beta - 1}} B \), respectively.

Note that \( \lambda_i' = \lambda_i^p \) and \( h_i' = h_i^p \), for \( \beta = 1 \), i.e., HRB-CUM and HPB-CUM are identical. Hence, we only focus on comparisons for \( \beta \geq 0 \) and \( \beta \neq 1 \) in the following sections.

3. ANALYSIS

In this section, we compare the corresponding hit probabilities \( h_i' \) and \( h_i^p \) and hit rates \( \lambda_i' \) and \( \lambda_i^p \), under these two approaches for different weights. Denote \( \Lambda = \sum_{i=1}^{n} \lambda_i \). Without loss of generality (w.l.o.g.), we assume that the arrival rates satisfy \( \lambda_1 \geq \cdots \geq \lambda_n \), such that the content popularities satisfy \( p_1 \geq \cdots \geq p_n \), where \( p_i = \lambda_i / \Lambda \). We first consider a general request process, and then consider the Zipf distribution, i.e., \( p_i = \frac{1}{i^\alpha} \) for \( i = 1, \cdots, n \), satisfying \( \sum_{i=1}^{n} p_i = 1 \). In particular, we choose the Zipf parameter \( \alpha = 0.8 \) with \( n = 10^3 \) and \( B = 100 \) in our numerical studies.

**Monotone decreasing weights**: Consider the case where the weights increase with request rate. W.l.o.g., we consider monotone decreasing weights, i.e., \( w_1 \geq \cdots \geq w_n \), given \( \lambda_1 \geq \cdots \geq \lambda_n \). Due to space constraints, we omit all the proofs, which are available in [4].

**Theorem 1.** When weights are monotone decreasing, (i) for \( \beta < 1 \), HRB-CUM favors more popular items compared to HPB-CUM, i.e., \( \exists j \in \{1, n\} \) s.t. \( h_i' > h_i^p \), \( \forall i < j \), and \( h_i' < h_i^p \), \( \forall i > j \); and (ii) for \( \beta > 1 \), HRB-CUM favors less popular items compared to HPB-CUM, i.e., \( \exists j \in \{1, n\} \) s.t. \( h_i' < h_i^p \), \( \forall i < j \), and \( h_i' > h_i^p \), \( \forall i > j \). In particular, if \( j, l \in \mathbb{Z}^+ \), then \( h_j' = h_l' \), and \( h_j^p = h_l^p \).

Figure 1: Hit Probability Comparison for \( \beta = 0.8 \) (Left) and \( \beta = 2 \) (Right)

The following corollary applies to the Zipf popularity distribution.

**Corollary 1.** Under the Zipf popularity distribution, we have the following results: (a) When \( \beta < 1 \), \( h_i' > h_i^p \) for \( i = 1, \cdots, i_0 \), and \( h_i' < h_i^p \), for \( i = i_0 + 1, \cdots, n \); (b) When \( \beta > 1 \), \( h_i' < h_i^p \), for \( i = 1, \cdots, i_0 \), and \( h_i' > h_i^p \), for \( i = i_0 + 1, \cdots, n \), where \( i_0 = \left\lfloor \frac{1}{\sum_{i} w_i^{1/\beta} \alpha(1 - \beta)} \right\rfloor \). In particular, we consider \( w_i = \lambda_i \) in our numerical studies.

The results are illustrated in Figure 1 for \( \beta = 0.8 \) and \( \beta = 2 \).

Now we are ready to make a comparison between the hit rate under these two approaches.

**Theorem 2.** When weights are monotone decreasing, (i) for \( \beta < 1 \), HRB-CUM favors more popular items compared to HPB-CUM, i.e., \( \exists j \in \{1, n\} \) s.t. \( \lambda_i' > \lambda_i^p \), \( \forall i < j \); and (ii) for \( \beta > 1 \), HRB-CUM favors less popular items compared to HPB-CUM, i.e., \( \exists j \in \{1, n\} \) s.t. \( \lambda_i' < \lambda_i^p \), \( \forall i < j \). In particular, \( \lambda_i' = \lambda_i^p \), for \( i = 1, \cdots, i_0 \), and \( \lambda_i' > \lambda_i^p \), for \( i = i_0 + 1, \cdots, n \), where \( i_0 = \left\lfloor \frac{1}{\sum_{i} w_i^{1/\beta} \alpha(1 - \beta)} \right\rfloor \). In particular, we consider \( w_i = \lambda_i \) in our numerical studies.

The following corollary applies to the Zipf popularity distribution.

**Corollary 2.** Under the Zipf popularity distribution, we have the following results: (a) When \( \beta < 1 \), \( \lambda_i' > \lambda_i^p \) for \( i = 1, \cdots, i_0 \), and \( \lambda_i' < \lambda_i^p \), for \( i = i_0 + 1, \cdots, n \); (b) When \( \beta > 1 \), \( \lambda_i' < \lambda_i^p \), for \( i = 1, \cdots, i_0 \), and \( \lambda_i' > \lambda_i^p \), for \( i = i_0 + 1, \cdots, n \), where \( i_0 = \left\lfloor \frac{1}{\sum_{i} w_i^{1/\beta} \alpha(1 - \beta)} \right\rfloor \). In particular, we consider \( w_i = \lambda_i \) in our numerical studies.

The results are illustrated in Figure 2 for \( \beta = 0.8 \) and \( \beta = 2 \).
4. ONLINE ALGORITHMS

In Section 2, we formulate the optimization problem with a fixed cache size, however, system parameters can change over time, and we need online algorithms to implement the optimal strategy to adapt to these changes in the presence of limited information. In the following, we develop such algorithms for HRB-CUM, and then make a comparison with the corresponding algorithms for HPB-CUM in [1]. Due to space limitations, we only focus on dual algorithms and skip the detailed proofs of all the results, which are available in [4].

**Dual Algorithm:** The HRB-CUM formulation in (3) is a convex optimization problem, and hence solving the dual problem will result in the optimal solution. Since we can easily ensure that $0 < t_1 < \infty$, then $0 < h'_i < 1$, i.e., $0 < \lambda'_i < \lambda_i$. Therefore, the Lagrange dual function is given as

$$D(\eta) = \max_{\lambda'_i} \left\{ \sum_{i=1}^{n} U_i(\lambda'_i) - \eta \left( \sum_{i=1}^{n} \frac{\lambda'_i}{\lambda_i} - B \right) \right\},$$

and the dual problem is $\min_{\eta \geq 0} D(\eta)$. Following the standard gradient descent algorithm by taking the derivative of $D(\eta)$ w.r.t. $\eta$, we have $\eta \leftarrow \max \left\{ 0, \eta + \gamma \left( \sum_{i=1}^{n} \frac{\lambda'_i}{\lambda_i} - B \right) \right\}$, where $\gamma > 0$ is the step size at each iteration and $\eta \geq 0$ due to KKT conditions.

Based on the results in Section 2, we have $\lambda'_i = U_i^{-1} \left( \frac{\eta}{\lambda_i} \right)$. Hit rates are then controlled by setting the timer $t_i$. As $\lambda'_i = \lambda_i(1 - e^{-\lambda_i t_i})$ for a reset TTL cache, we can express $t_i$ in terms of $\eta$, and the dual algorithm for reset TTL cache can be summarized as

$$t_i = \frac{-1}{\lambda_i} \log \left( 1 - \frac{1}{\lambda_i} U_i^{-1} \left( \frac{\eta}{\lambda_i} \right) \right),$$

$$\eta \leftarrow \max \left\{ 0, \eta + \gamma \left( \sum_{i=1}^{n} \frac{\lambda'_i}{\lambda_i} - B \right) \right\}.$$ \hspace{1cm} (6a) \hspace{1cm} (6b)

Denote the optimal value for $\eta$ as $\eta^*$, then with a proper Lyapunov argument, we can show that the above dual algorithm converges to the optimal solution.

**Comparisons:** In the following, we implement the dual algorithm for HRB-CUM and compare it to that for HPB-CUM [1]. Due to space restrictions, we limit our study to minimum potential delay fairness, i.e., $\beta = 2$.

In our experiments, we consider a cache size $B = 100$ serving $n = 1000$ contents, where $\eta$ is updated according to (6b). Requests arrive according to a Poisson process with aggregate rate $\Lambda = 1$. We assume that content popularities follow a Zipf distribution with parameter $\alpha = 0.8$.

**Convergence Rate and Robustness:** As discussed earlier, the dual algorithm is globally and asymptotically stable, and converges to the optimal solution. However, it says nothing about how fast it converges. From (6b), it is clear that the step size $\gamma$ plays a significant role in the convergence rate. We choose different values of $\gamma$ to compare the performance of HRB-CUM and HPB-CUM, shown in Figure 3. On one hand, we find that when a larger value of $\gamma = 10^{-3}$ is chosen, the dual algorithm of HRB-CUM easily converges after a few number of iterations, i.e., the simulated hit probabilities exactly match numerically computed values, while that of HPB-CUM does not converge. On the other hand,

![Figure 3: Hit Probability and Cache size distribution comparisons for online dual algorithm with $\gamma = 10^{-3}$ (Left) and $\gamma = 10^{-5}$ (Right)](image)

when a smaller value $\gamma = 10^{-5}$ is chosen, both converge under the same number of iterations. We have also used $\gamma = 10^{-1}, 10^{-7}$, which exhibit similar phenomenon to $10^{-3}$ and $10^{-5}$, respectively, and hence are omitted due to space constraints. Furthermore, we have explored the expected number of contents in the cache, shown in Figure 3. It is obvious that under the HRB-CUM, the probability of violating the target cache size $B$ is quite small, while that probability for HPB-CUM is large especially for $\gamma = 10^{-3}$, and even for $\gamma = 10^{-5}$, HRB-CUM is more concentrated on the target size $B$. These results indicate that the dual algorithm associated with HRB-CUM is more robust to changes of the step size parameter $\gamma$ and converge to the optimal values much faster.

**Remark 2:** We have also characterized and implemented the online primal and primal-dual algorithms. Under both cases, HRB-CUM achieves better or similar performance compared to HPB-CUM. Moreover, for proportional-fairness $\beta = 1$ and min-max fairness $\beta = \infty$, given (4), we can easily check that they are equivalent to the results in [1], hence are omitted here.

**Discussion:** In this paper, we have proposed a HRB-CUM, developed decentralized online algorithms to implement the optimal policies and characterized the advantages of HRB-CUM over HPB-CUM. Further study can be done on exploring online algorithms w.r.t. estimation of content request rates. Also non-reset TTL might have different implications on the design and performance of these algorithms.

5. REFERENCES

LRU Cache under Stationary Requests

Bo Jiang  
College of Information and Computer Sciences  
University of Massachusetts  
Amherst MA 01003, USA  
bjiang@cs.umass.edu

Philippe Nain  
Inria  
Ecole Normale Supérieure de Lyon, LIP, 46 allée d’Italie  
69364 Lyon, France  
philippe.nain@inria.fr

Don Towsley  
College of Information and Computer Sciences  
University of Massachusetts  
Amherst MA 01003, USA  
towsley@cs.umass.edu

ABSTRACT
In this paper we extend an approximation first proposed by Fagin [4] for the LRU cache under the independence reference model to systems where requests for different contents form independent stationary and ergodic processes. We show that this approximation becomes exact as the number of contents goes to infinity while maintaining the fraction of the contents that can populate the cache to be constant. Last, we provide results on the rate of convergence.

Keywords
Cache, LRU, Characteristic time approximation, Stationary request processes

1. INTRODUCTION
Caches form a key component of many computer networks and systems. Moreover they are becoming increasingly more important with the current development of new content-centric network architectures. A variety of cache replacement algorithms has been introduced and analyzed over the last few decades, most based on the least recently used algorithm (LRU). Considerable work has focused on analyzing these policies. However even the simple LRU policy defies exact analysis that leads to computationally tractable results. This has led instead to the development of accurate approximations, [4, 3, 2].

The most useful approximation for LRU was introduced by Fagin in [4] for the independent reference model (IRM). Briefly, Fagin introduced the concept of a characteristic time (our terminology) and showed asymptotically that the performance of LRU converges to that of a TTL cache with a timer set to the characteristic time. This work disappeared and several papers [2, 5] reintroduced this approximation. More recently, [6] extended the characteristic time (CCT) approximation to a setting where requests for distinct contents are independent and described as renewal processes. The accuracy of this approximation is supported by simulations but a theoretical basis is lacking. Providing a theoretical justification of this extended CCT approximation is the focus of this paper.

The main contribution of this paper is to extend Fagin’s results for LRU under IRM assumptions to the more general setting where requests for different content are independent of each other but requests to each content are described by a stationary and ergodic process. Based on these results, we develop a CCT approximation for the performance of an LRU cache. Furthermore, we develop bounds on the error introduced by the CCT approximation along with a convergence rate of the approximation to the asymptotic limit as the cache size and number of contents increase to infinity.

The rest of the paper is organized as follows. Section 2 presents our model of an LRU cache under a general request model. Section 3 presents the extension to Fagin’s result to the case where requests for contents are described by independent stationary and ergodic processes. Section 4 investigates the rate of convergence of the LRU cache hit probability to the TTL cache hit probability. Last, concluding statements are provided in Section 5.

2. MODEL
We consider a cache of size $C$ serving $n$ unit size contents labelled $i = 1, \ldots, n$ where $C \in (0, n)$. Requests for the contents are described by $n$ independent stationary and ergodic point processes $N_i := \{t_i(k), k \in \mathbb{Z}\}$, where $-\infty < t_i(-1) < t_i(0) \leq 0 < t_i(1) < \cdots < \infty$ represent the successive request times to content $i = 1, \ldots, n$ having probability measure $\mathbb{P}$. Let $0 < \lambda_i < \infty$ denote the intensity of request process $N_i$, i.e., the long term average request rate for content $i$ (see e.g. [1, Sections 1.1 and 1.6] for an introduction to stationary and ergodic point processes).

Let $\mathbb{P}^0$ be the Palm probability associated with the point process $N_i$ (see e.g. [1, Eq. (1.2.1)]). In particular, $\mathbb{P}^0\{t_i(0) = 0\} = 1$. In other words, under $\mathbb{P}^0$ content $i$ is requested at time $t = 0$. Define $G_i(x) = \mathbb{P}^0\{t_i(1) \leq x\}$, the cdf of the duration between two successive requests to content $i$ under $\mathbb{P}^0$. It is known that $\mathbb{E}^0\{t_i(1)\} = 1/\lambda_i$ [1, Exercise 1.2.1], with $\mathbb{E}^0$ the expectation operator associated with $\mathbb{P}^0$.

Last, we define $\mathbb{P}^0$, the Palm probability associated with the point process $\{t(k), k \in \mathbb{Z}\}$, $-\infty < t(-1) < t(0) \leq 0 < t(1) < \cdots < \infty$, resulting from the superposition of the $n$ independent point processes $N_1, \ldots, N_n$. Under $\mathbb{P}^0$ a content is requested at time $t = 0$ (since $\mathbb{P}^0\{t(0) = 0\} = 1$). Let $X_0 \in \{1, \ldots, n\}$ denote this content. We denote by $\mathbb{E}^0$ the expectation operator associated with $\mathbb{P}^0$. It is known that (see e.g. [1, Section 1.4.2])

$$
\mathbb{P}^0\{X_0 = i\} = \frac{\lambda_i}{\Lambda(n)} := p_i^{(n)},
$$

(1)

Copyright is held by author/owner(s).
with \( \Lambda^{(n)} := \sum_{i=1}^{n} \lambda_i \).
For any cdf \( F \) with support in \([0, \infty)\), let
\[
\hat{F}(x) = \frac{1}{EF} \int_{0}^{x} \hat{F}(y)dy,
\]
where \( \hat{F} = 1 - F \) is the ccdf, and \( EF = \int_{0}^{\infty} F(y)dy \) is the mean. It is well-known that (see e.g. [1, Section 1.3.4])
\[
P[-t_i(0) \leq x] = P[t_i(1) \leq x] = \lambda_i \int_{0}^{x} \hat{G}_i(y)dy = \hat{G}_i(x)
\]
for each \( i \). We assume that
\[
G_i(x) = G(\lambda_i x),
\]
for some cdf \( G \) with mean 1. Note that (4) holds if \( G(\cdot) \) is the exponential distribution. It follows from (2) and (4) that
\[
\hat{G}_i(x) = \hat{G}(\lambda_i x).
\]
We also assume that there exists a continuously differentiable cdf \( F \) with support in \([0, 1] \) such that for \( i = 1, 2, \ldots, n \)
\[
p'_i(n) = F\left(\frac{i}{n}\right) - F\left(\frac{i-1}{n}\right) = \frac{1}{n} F'(\xi_i(n)),
\]
where \( \xi_i(n) \in \left(\frac{i-1}{n}, \frac{i}{n}\right) \). The existence of \( \xi_i(n) \) is guaranteed by the mean-value theorem. We assume that \( F'(x) > 0 \) a.e. on \([0, 1] \). We allow \( F'(0) \) to be infinite, to allow Zipf’s law in particular.

Let \( Y_i(t) = 1 \) if content \( i \) was requested during the interval \([-t, 0) \) and \( Y_i(0) = 0 \) otherwise. With this notation, \( Y(t) := \sum_{i=1}^{n} Y_i(t) \) is the number of distinct contents requested during \([-t, 0) \). Let \(-\tau, 0) \) be the smallest past interval such that there have been \( C \) distinct contents referenced in that interval, i.e.,
\[
\tau = \inf\{t : Y(t) \geq C\}.
\]
Note that if we reverse the arrow of time, we obtain in steady-state statistically the same request processes, and \( \tau \) is a stopping time for the process \( Y(t) \). The stationary hit probability of an LRU cache is then given by
\[
H_{LRU}^{TTL} = \mathbb{P}[Y_{n_0}(\tau) = 1].
\]
If the cache is a TTL cache with timer \( T \), the stationary hit probability is
\[
H_{TTL}(T) = \mathbb{P}[Y_{n_0}(T) = 1].
\]
Note that \( \mathbb{P}[Y_i(\tau) = 1] \) and \( \mathbb{P}[Y_i(T) = 1] \) are the stationary hit probabilities of content \( i \) in an LRU cache and in a TTL cache with timer \( T \), respectively. Define
\[
\beta^*(\nu) = \int_0^1 \hat{G}(vF'(x))dx, \quad h^*(\nu) = \int_0^1 F'(x)G(vF'(x))dx.
\]
The next section shows that, as \( n \) becomes large, an LRU cache behaves as a TTL cache with a timer value that we identify.

3. ASYMPTOTIC BEHAVIOR
Throughout \( T_n(\nu) = n\nu/\Lambda^{(n)} \) and \( \beta_0 \in (0, 1) \).

**Proposition 1.** Assume that \( C \sim n\beta_0 \). Then,
\[
\lim_{n \to \infty} H_{LRU}^{TTL}(T_n(\nu_0)) = h^*(\nu_0),
\]
where \( \nu_0 \) is the unique solution in \((0, \infty)\) of \( \beta^*(\nu) = \beta_0 \). If \( G \) is continuous then, as \( n \to \infty \),
\[
\max_{1 \leq i \leq n} \left[ \mathbb{P}^i[Y_i(\tau) = 1] - \mathbb{P}^i[Y_i(T_n(\nu_0)) = 1]\right] \to 0.
\]

This result states that the hit probability of a LRU cache converges to that of a TTL cache with timer \( T \sim n\nu_0/\Lambda^{(n)} \) as the cache size and number of contents increase to infinity. It was first proved rigorously by Fagin [4] in the IRM setting. Proposition 1 provides a rigorous extension of Fagin’s result to independent and stationary request processes. It is easy to extend this proposition to several content popularity cdfs. The latter setting is useful when, for instance, several service providers share a common LRU cache and that contents associated with different providers exhibit different popularity probability distributions - see [7] for details.

The proof of Proposition 1 relies on the three following lemmas whose proofs are found in [7].

**Lemma 1.** The equation \( \beta^*(\nu) = \beta_0 \) has a unique solution \( \nu = \nu_0 \) in \((0, \infty)\).

The next lemma focuses on the hit probability in a TTL cache with timer \( T \) as the number of contents \( \to \infty \).

**Lemma 2.** For \( \nu > 0 \),
\[
\lim_{n \to \infty} H_{TTL}(T_n(\nu)) = h^*(\nu).
\]

**Lemma 3.** For \( T > 0 \),
\[
\mathbb{P}[Y_{n_0}(\tau) = 1, \tau \leq T] \leq \mathbb{P}[Y_{n_0}(T) = 1, \tau \leq T] \leq \mathbb{P}[Y_{n_0}(T) = 1, \tau > T] \geq \mathbb{P}[Y_{n_0}(\tau) = 1, \tau \leq T].
\]

The next lemma shows that \( \tau \) is concentrated around \( T_n(\nu_0) \).

**Lemma 4.** Assume that \( C \sim \beta_0 n \). For \( \nu_1 < \nu_0 < \nu_2 \),
\[
\lim_{n \to \infty} \mathbb{P}[\tau < T_n(\nu_1)] = \lim_{n \to \infty} \mathbb{P}[\tau > T_n(\nu_2)] = 0.
\]

We are now in position to prove Proposition 1.

**Proof of Proposition 1.** Let \( \nu_1 < \nu_0 < \nu_2 \). We have
\[
H_{LRU}^{TTL} = \mathbb{P}[Y_{n_0}(\tau) = 1] \geq \mathbb{P}[Y_{n_0}(\tau) = 1, \tau \geq T_n(\nu_1)] \\
\geq \mathbb{P}[Y_{n_0}(T_n(\nu_1)) = 1, \tau \geq T_n(\nu_1)] \text{ by Lemma 3} \\
\geq \mathbb{P}[Y_{n_0}(T_n(\nu_1)) = 1] - \mathbb{P}[\tau < T_n(\nu_1)] \\
= H_{TTL}(T_n(\nu_1)) - h^*(\nu) \\
\]

With the help of Lemmas 2 and 4 we find
\[
\liminf_{n \to \infty} H_{LRU}^{TTL} \geq \lim_{n \to \infty} H_{TTL}(T_n(\nu_1)) = h^*(\nu_1).
\]

Letting \( \nu_1 \to \nu_0 \) gives \( \liminf_{n \to \infty} H_{LRU} \geq h^*(\nu_0) \). For the other direction, note that
\[
H_{LRU}^{TTL} = \mathbb{P}[Y_{n_0}(\tau) = 1] \\
\leq \mathbb{P}[Y_{n_0}(\tau) = 1, \tau \leq T_n(\nu_2)] + \mathbb{P}[\tau > T_n(\nu_2)] \\
\leq \mathbb{P}[Y_{n_0}(T_n(\nu_2)) = 1, \tau \leq T_n(\nu_2)] + \mathbb{P}[\tau > T_n(\nu_2)] \\
\leq \mathbb{P}[Y_{n_0}(T_n(\nu_2)) = 1] + \mathbb{P}[\tau > T_n(\nu_2)] \\
= H_{TTL}(T_n(\nu_2)) + \mathbb{P}[\tau > T_n(\nu_2)],
\]
where (11) follows from Lemma 3. With the help of Lemmas 2 and 4 we obtain
\[
\limsup_{n \to \infty} H_{LRU}^{TTL} \leq \lim_{n \to \infty} H_{TTL}(T_n(\nu_2)) = h^*(\nu_2).
\]
Letting $\nu_2 \to \nu_0$ yields $\limsup_n H^{\text{LRU}} < h^*(\nu_0)$. Therefore $\lim_{n \to \infty} F^{\text{LRU}} = h^*(\nu_0)$. The proof of the uniform convergence result (10) can be found in \cite{7}.

4. RATE OF CONVERGENCE

In this section we investigate the rate of convergence of $H^{\text{TTL}}$ to $H^{\text{LRU}}$ in $T_0$, where the timer $T_0$ is defined below.

\begin{equation}
\Lambda(n) = 1.\text{ Define } C(T) = \mathbb{E}[Y(T)], \text{ the expected number of contents in a TTL cache with timer } T. \end{equation}

\begin{equation}
\text{LEMMA } 5. \ C(T) = \sum_{i=1}^{n} G_i(T) \text{ and } T \to C(T) \text{ is strictly increasing for all } T < n. \end{equation}

From $C(0) = 0$, $C(\infty) = n$ and the strict increasingness of $T \to C(T)$, we know that $C(T) = C \in (0, n)$ has a unique solution in $(0, \infty)$, which we call $T_0$. Define $\mu_0 = C(T_0) = \sum_{i=1}^{n} \lambda_i(1-G_i(T_0))$, the miss rate in a TTL cache with timer $T_0$. Assume that $\mu_0 > 0$. Let $\delta_1 = 0$ if the request process for content $i = 1, \ldots, n$ is Poisson and $\delta_1 = 1$ otherwise.

\begin{equation}
\text{PROPOSITION 2. Let } \delta = \max_{1 \leq i < n} \delta_i \text{ and } \mu \in (0, \mu_0). \text{ Suppose there exist a constant } B \text{ and } \rho \in \left( \frac{1}{\mu_0}, \mu_0 \right] \text{ such that }
\end{equation}

\begin{equation}
|G_i(T_0) - G_i(T_0 \pm \delta_0)| \leq \frac{B}{\rho} \text{ for } \rho \in \left[ \frac{\delta}{\mu_0}, \mu_0 \right]. \tag{12} \end{equation}

Let $D_0 = \sqrt{\frac{3}{2}} \mu T_0$. Assume $D_0/B \geq \sqrt{2}$ and

\begin{equation}
1 + \sqrt{\frac{\log D_0}{B}} \leq D_0 \rho - \sqrt{\frac{2}{\rho}}. \end{equation}

Then,

\begin{equation}
\Delta_n(T_0) := |H^{\text{LRU}} - H^{\text{TTL}}(T_0)| 
\leq \max_{1 \leq i \leq n} \left| P_i^0[Y_i(\tau) = 1] - P_i^0[Y_i(T_0) = 1]\right|
\leq \frac{B \delta}{\mu_0} + \frac{B}{D_0} \left( \sqrt{\frac{\log D_0}{B}} + 1 + \sqrt{2} \right). \end{equation}

See \cite{7} for the proof. Note that (12) holds for a large class of distributions.

\begin{example}
For Poisson arrivals, $G_i(t) = e^{-\lambda_i t}$. For any $\varepsilon \geq 0$,
\begin{align*}
0 \leq G_i(T_0 - \varepsilon) - G_i(T_0) &\leq e^{\lambda_i \varepsilon} (1 - e^{-\lambda_i \varepsilon}) \\
&\leq \lambda_i T_0 e^{-\lambda_i T_0} \leq e^{-\varepsilon}. \end{align*}
\end{example}

For $\varepsilon \in [0, 1]$, $\varepsilon 
\begin{align*}
0 \leq G_i(T_0 - \varepsilon) - G_i(T_0) &\leq \sup_{\varepsilon \geq 0} e^{-\varepsilon} (1 - e^{\varepsilon}) \\
&= (1 - e)^{\frac{1}{1 - e}} \leq \varepsilon. \end{align*}

Thus (12) holds with $B = 1$ and $\rho = \mu_0 - \mu$.

\begin{example}
Suppose $G_i$’s are continuously differentiable. By the mean value theorem, there exists $\xi_i \in [1, 1 + \varepsilon]$ such that
\begin{align*}
0 \leq G_i(T_0) - G_i(T_0 + \varepsilon) &\leq G_i'(\xi_i T_0) \varepsilon \\
&\leq \xi_i T_0 G_i'(\xi_i T_0) \varepsilon \leq \sup_{t \geq 0} G_i'(t) \varepsilon.
\end{align*}

Similarly, there exists $\xi_i \in [1 - \varepsilon, 1]$ such that
\begin{align*}
0 \leq G_i(T_0 - \varepsilon) - G_i(T_0) &\leq \sup_{t \geq 0} G_i'(t) \varepsilon
\end{align*}

\begin{align*}
\leq \frac{\xi_i}{1 - \varepsilon} T_0 G_i'(\xi_i T_0) \varepsilon \leq \frac{\varepsilon}{1 - \rho} \left( \sup_{t \geq 0} G_i'(t) \right).
\end{align*}

Thus (12) holds with $\rho \in \left[ \frac{\delta}{\mu_0} \gamma, \frac{\mu_0 - \mu}{\mu_0} \right]$ and $B = \frac{A}{1 - \rho}$.

Below is the rate of convergence for Poisson arrivals.

\begin{corollary}
Assume that $\delta_i = 0$ for each $i$. If $C \sim \beta_0 n$, $\min_i \lambda_i = \Omega(n^{-\gamma})$ for $1 \leq \gamma < 3/2$,
\begin{align*}
\Delta_n(T_0) &\leq \max_{1 \leq i \leq n} \left| P_i^0[Y_i(\tau) = 1] - P_i^0[Y_i(T_0) = 1]\right| \\
&= O \left( \frac{n^{3/2}}{\sqrt{\log n}} \right).
\end{align*}

5. CONCLUSIONS

In this paper, we developed an approximation for the aggregate and individual content hit rates of an LRU cache for the case that content requests are described by independent stationary and ergodic processes. This approximation extends one first proposed and studied by Fagin \cite{4} for the independent reference model and provides the theoretical basis for approximations introduced in \cite{6} for content requests described by independent renewal processes. We showed that the approximations become exact in the limit as the number of contents goes to infinity while the ratio of this and the cache size remains constant. Last, we established the rate of convergence for the approximation as number of contents increases.

Future directions include extension of these results to other cache policies such as FIFO and random and to networks of caches. In addition, it is desirable to relax independence between different content request streams.

6. REFERENCES

\begin{itemize}
\item [1] F. Baccelli and P. Brémaud. Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recursions, volume 26 of Applications of Mathematics, Stochastic Modelling and Applied Probability. \cite{1}
\end{itemize}
Thinking Fast and Slow: Optimization Decomposition Across Timescales

Gautam Goel  Niangjun Chen  Adam Wierman

ABSTRACT

Many real-world control systems, such as the smart grid and software defined networks, have decentralized components that react quickly using local information and centralized components that react slowly using a more global view. This work seeks to provide a theoretical framework for how to design controllers that are decomposed across timescales in this way. The framework is analogous to how the network utility maximization framework uses optimization decomposition to distribute a global control problem across independent controllers, each of which solves a local problem; except our goal is to decompose a global problem temporally, extracting a timescale separation. Our results highlight that decomposition of a multi-timescale controller into a fast timescale, reactive controller and a slow timescale, predictive controller can be near-optimal in a strong sense. In particular, we exhibit such a design, named Multi-timescale Reflexive Predictive Control (MRPC), which maintains a per-timestep cost within a constant factor of the offline optimal in an adversarial setting.

1. INTRODUCTION

Modern control systems nearly always operate at multiple timescales. In the power grid, slow timescale economic dispatch is used to determine which baseload generators will supply power, while fast timescale frequency regulation is used to correct any imbalance between demand and supply that may arise [1]. In networking, software defined networks use a slow timescale “control plane” controller to decide where to send data packets, whereas fast timescale “data plane” controllers are responsible for routing the actual data [4].

Thus, the design and analysis of multi-timescale control systems has received considerable attention. However, the design of control policies for multi-timescale control systems typically does not address the joint problem of designing control policies across timescales. Instead, controllers for each timescale are designed independently. For example, in the power grid, the slow timescale problem of economic dispatch is usually studied separately from the fast timescale problem of frequency regulation. Recent work jointly designing economic dispatch and frequency regulation in the the power grid highlights significant inefficiency in designs that treated the two timescales independently [1].

In this work, our goal is to develop a framework for deriving rather than assuming a timescale separation in global optimization problems. In particular, we adapt the idea of optimization decomposition from the domain of distributed control into the domain of multi-timescale control.

There is a vast literature on optimization decomposition, and fields as diverse as Internet congestion control [5,8], smart grid control [1], and beyond [2]. The idea of this approach is to decompose a global optimization problem into smaller localized subproblems, each of which is solved by independent controllers. See [2] for a survey. In a similar way, our goal in this work is to look for decompositions of a global optimization problem in time, as opposed to in space.

However, this goal is made challenging by the tight coupling between the timescales due to the underlying dynamics of the system under consideration. Typically, spatial optimization decomposition is performed for static optimizations, but in multi-timescale control the dynamics of the system cannot be ignored. Any slow timescale action will impact the future state via the dynamics and hence must be taken into account when designing the fast controller; conversely, any fast timescale action impacts the state seen by a slow controller and thus impacts its design as well. This makes it unclear whether it is possible to achieve a clean separation between controllers at different timescales.

We make two main contributions in this work. We first introduce a simple but general model for studying multi-timescale optimal control. We consider a system subject to linear dynamics which is perturbed by noise; we make absolutely no assumptions about the nature of the noise, i.e., it may be random or even adversarial. This system can be controlled by two controllers, one of which is a traditional, “fast timescale” controller that can react immediately to the noise, and another which is a novel, “slow timescale” controller that is only able to react slowly, but which is empowered with access to more information than the fast controller and is potentially cheaper to use.

We then introduce a new multi-timescale control policy, MRPC, and derive strong guarantees on its performance. In particular, the per-step cost incurred by our algorithm is at most a constant more than that incurred by the offline optimal. The design of our policy is motivated by a
structural result about the offline optimal control action, which highlights a strong decomposition between fast and slow timescale controllers. Applying this idea to the design of the online algorithm, we are able to achieve a clean separation between timescales. Remarkably, our decomposition results in a purely reflexive, “dumb” fast controller, which performs no optimization or lookahead. Thus, all of the computational burden is shifted onto the slow, “smart” controller. This property of MRPC is desirable in many applications since the slow controller is often centralized and able to take a global view of the system, but the fast controllers are decentralized and myopic, e.g., the power systems and networking examples mentioned above.

2. MODEL

Our goal is to study the design of controllers for systems that operate at multiple timescales. To this end, we focus on a simple but general optimal control problem.

The multi-timescale problem we consider builds on the following optimal control problem, which operates at a single timescale:

\[
\min_{x_f,s} \sum_{t=1}^{T} c_x(x_t) + c_f(f_t) + c_s(s_t) \quad \text{(1)}
\]

\[
\text{s.t. } \begin{align*}
   x_t &= Ax_{t-1} + B^f f_t + B^s s_t + w_t \\
   x_0 &= 0 \\
   s_t &= s_{t-1} \quad \forall t \notin \{0, 2k, 2k+1, \ldots\}
\end{align*}
\]

Here \(x_t \in \mathbb{R}^n\) is the state variable, \(f_t \in \mathbb{R}^n\) is the control action and \(w_t \in \mathbb{R}^n\) is the disturbance. In our technical results, we assume that the control matrix \(B^f\) is invertible; considering the non-invertible case is an interesting direction for future work. The cost functions \(c_x(\cdot), c_f(\cdot)\) are usually assumed to be non-negative and convex. The special case when each noise increment \(w_t\) is an i.i.d. Gaussian random variable and \(c_x(\cdot), c_f(\cdot)\) are positive definite quadratic forms represents the Linear Quadratic Regulator (LQR) framework [7].

To extend (1) to a multi-timescale control problems, we introduce a “slow” controller. The slow controller reacts much less quickly to noise than the fast controller; however there are two potential benefits afforded by the existence of the slow controller.

First, in many situations the slow controller is centralized, and hence can use global information to make better decisions than the decentralized, localized fast controllers. In our context, we model this by allowing the slow controller access to predictions of future noise increments. An example where the slow controller has this benefit is software defined networking, where the centralized controller has access to a limited number of noisy predictions of future noise increments. In particular, we assume that, at the start of each slow timescale interval, we have estimates \(\hat{w}_t\) of the true noise increments over that slow timescale interval. Importantly, we do not make distributional assumptions about the predictions or prediction errors. In the full version of this work, we use a reduction from online convex optimization to argue that predictions are in fact necessary [3].

3. ARCHITECTURAL DECOMPOSITION FOR MULTI-TIMESCALE CONTROL

We now turn our attention to the joint multi-timescale control problem in (2), and focus on the co-design of fast and slow controllers. Recall that, while the slow controller cannot act as frequently, there are two benefits it usually provides: (i) it may have more information and computational power than the fast controller, e.g., in software defined networking and robotics, and (ii) it may be cheaper to operate than the fast controller, e.g., when scheduling generation in the smart grid. To capture these benefits of a slow controller, we consider a setting where the slow controller has access to noisy predictions but the fast controller does not. We also specifically highlight the case where the slow controller is cheaper to operate, though our results apply more generally.

Our main result in this section provides a performance bound for a new, near-optimal algorithm – Multi-timescale Reflexive Predictive Control (MRPC) – that consists of a simple, reflexive fast timescale controller and a predictive slow timescale controller. For concreteness and ease of presentation we focus on the case where the cost functions \(c_x, c_s, c_f\) in (2) are norms \(\| \cdot \|_x, \| \cdot \|_s, \| \cdot \|_f\). Instead of merely presenting MRPC, we show that it can be derived naturally from understanding the structure of the offline optimal solution to (2).
The first step in the analysis is to establish a lower bound on the cost incurred by the offline optimal. As mentioned above, this lower bound highlights the decomposition between fast and slow used in the design of MRPC: We defer all proofs to the full version [3].

Lemma 1. Letting $\text{OPT}$ denote the optimal solution for (2), we have:

$$OPT \geq \min_s \sum_{r \in S} \left[ k\|s_r\|_s + C \sum_{t=r}^{r+k-1} \|(B^f)^{-1}(B^s s_t + w_t)\|_f \right]$$

where

$$C = \min \left( \frac{c}{2\|(B^f)^{-1}\|_1} \right)$$

and $c$ is a constant such that $\|v\|_s \geq c\|v\|_f$ for all $v$.

The lower bound has the following interpretation. Suppose the state is set to zero. After the slow controller has set its action to be $s_r$, the fast control action which corrects the deviation from zero is $(B^f)^{-1}(B^s s_r + w_t)$, and our lower bound is the sum of the resulting costs (up to the constant $C$). Notice that the fast controller is extremely simple: it uses no predictions and performs no optimization. All of the predictions and optimizations are shifted onto the slow controller. As the true noise increments $w_t$ are revealed one by one, the fast controller myopically corrects any noise so as to keep the state at zero.

Formally, let $\hat{f}$ and $\hat{s}$ denote the fast and slow control actions of MRPC. Then, the operation of each is as follows:

$$\hat{s}_r = \min_s \left[ k\|s_r\|_s + \sum_{t=r}^{r+k-1} \|(B^f)^{-1}(B^s s_t + \hat{w}_t)\|_f \right] \quad (3)$$

$$\hat{f}_t = -(B^f)^{-1}(B^s \hat{s}_r + w_t) \quad t = r, \ldots, r + k - 1 \quad (4)$$

Notice that the fast controller is very simple; it uses no predictions and performs no optimization. All of the prediction and optimization is shifted onto the slow controller. This is consistent with how the two controllers are used in many applications, where the slow controller is often centralized, with access to global information, but the fast controllers are usually decentralized, localized, and computationally limited. For example, in the smart grid a slow timescale global optimization problem is solved (economic dispatch) and then localized fast timescale controllers myopically correct any deviations that may arise (frequency regulation).

The following lemma provides an upper bound on the cost of MRPC:

Lemma 2.

$$\text{MRPC} \leq \min_s \sum_{r \in S} \left[ k\|s_r\|_s + \sum_{t=r}^{r+k-1} \|(B^f)^{-1}(B^s s_t + w_t)\|_f \right] + \|(B^f)^{-1}\| \sum_{t=1}^{T} \|\hat{w}_t - w_t\|_f$$

Together, our two lemmas immediately give a strong performance bound on MRPC:

Theorem 1. Assume the cost functions $c_x, c_f, c_y$ in (2) are norms $\|\cdot\|_x, \|\cdot\|_f$, $\|\cdot\|_y$. Then MRPC has an average per-stage cost within a constant factor of optimal. Specifically,

$$\frac{\text{MRPC}}{T} \leq \max \left( \frac{2\|(B^f)^{-1}\|_1}{c} \frac{\text{OPT}}{T} + 2\|(B^f)^{-1}\| E(\hat{w}, w) \right)$$

where $c$ is a constant such that $\|v\|_s \geq c\|v\|_f$ for all $v$ and $E(\hat{w}, w)$ is the sample path average prediction error:

$$E(\hat{w}, w) = \frac{1}{T} \sum_{t=1}^{T} \|\hat{w}_t - w_t\|_f$$

To get intuition for the bound itself, let us first look at the second term. The second term in the bound corresponds to the inefficiency due to noisy predictions. In particular, if we assume perfect lookahead (i.e. $\hat{w}_t = w_t$ for all $t$), then the second term disappears. Thus, we see that prediction error has only an additive effect. It is important to realize that our analysis makes no modeling assumptions on the form of the prediction error. The error can be adversarial or stochastic and the result still holds.

The first term bounds the per-step cost incurred by our algorithm relative to the per-step cost incurred by the offline optimal. To get intuition for it, consider the case where control costs dominate the state costs. Specifically, consider the case where $c \geq 2\|(B^f)^{-1}\|_1$, and there are no errors in predictions. In this case, we have $\text{MRPC} = \text{OPT}$. It is worth highlighting this result in words: when state costs dominate control costs and prediction errors are small, our distributed algorithm achieves the optimal value of (2).

4. REFERENCES


Distributed Optimization via Local Computation Algorithms

Palma London  Nianjun Chen  Shai Vardi  Adam Wierman

ABSTRACT

We propose a new approach for distributed optimization based on an emerging area of theoretical computer science—local computation algorithms. The approach is fundamentally different from existing methodologies and provides a number of benefits, such as robustness to link failure and adaptivity in dynamic settings. Specifically, we develop an algorithm, LOCO, that given a convex optimization problem \( P \) with \( n \) variables and a “sparse” linear constraint matrix with \( m \) constraints, provably finds a solution as good as that of the best online algorithm for \( P \) using only \( O((n + m)) \) messages with high probability. The approach is not iterative and communication is restricted to a localized neighborhood. In addition to analytic results, we show numerically that the performance improvements over classical approaches for distributed optimization are significant, e.g., it uses orders of magnitude less communication than ADMM.

1. INTRODUCTION

The goal of this work is to introduce a new, fundamentally different approach to distributed optimization based on an emerging area of theoretical computer science—local computation algorithms (LCAs) [5, 8, 10, 11].

There are a wide variety of approaches for distributed optimization, which fall into the categories of dual decomposition and subgradient methods, and consensus-based schemes. While these approaches are distributed, they are not local. A local algorithm is one where a query about a small part of a solution to a problem can be answered by communicating with only a small neighborhood around the part queried. Neither iterative descent methods nor consensus methods are local because answering a query about a piece of the solution requires global communication.

This work introduces an algorithm, LOCO, (LOCal Convex Optimization) that is both distributed and local. It is not an iterative method and uses far less communication to compute the solution than iterative descent and consensus methods. While the technique we propose is general, in this work we focus on a canonical optimization problem: network utility maximization. We provide worst-case guarantees on the performance of LOCO with respect to the relative error and the number of messages it requires.

The key idea behind LOCO is an extension of recent results on LCAs; this work, for the first time, imports techniques from the field of local computation algorithms into the domain of networked control. In particular, a key insight is that online algorithms can be converted into local algorithms in graph problems with bounded degree [8]. The technical contribution of this work is the extension of these ideas to convex programs.

Local computation methods are well suited for distributed optimization. For example, any failure in the system only has local effects: if a node in a distributed system goes offline while an iterative distributed algorithm is executing, the whole process is brought to a halt; if the computations are all local, the failure will only affect a small number of nodes in the neighborhood of the failure. Similarly, lag in a single edge affects the computation of the entire solution in the iterative setting, while most computations are not affected when the computations are local. Another advantage of local computation is that it allows the system to be more dynamic: an arrival of another node requires recomputing the entire solution if the algorithm is not local, but requires only a few local messages and computations if the algorithm is local.

2. NETWORK UTILITY MAXIMIZATION

To illustrate the application of local computation algorithms to distributed optimization, we focus on the classic setting of network utility maximization (NUM). Consider a network containing a set of links \( \mathcal{L} = \{1, \ldots, m\} \) of capacity \( c_j \), for \( j \in \mathcal{L} \). A set of \( N = \{1, \ldots, n\} \) sources share the network; source \( i \in N \) is characterized by \( (L_i, f_i, \underline{x}_i, \bar{x}_i) \): a path \( L_i \subseteq \mathcal{L} \) in the network; a concave utility function \( f_i : \mathbb{R}_+ \rightarrow \mathbb{R} \); and the minimum and maximum transmission rates of \( i \). The optimization of aggregate utility can be formulated as follows,

\[
\max_x \sum_{i=1}^{n} f_i(x_i)
\]

subject to \( Ax \leq c, \underline{x} \leq x \leq \bar{x}, \)

where \( A \in \mathbb{R}^{m \times n} \) is defined as \( A_{ij} = 1 \) if \( j \in L_i \) and 0 otherwise. The goal in NUM is to maximize the sources’ aggregate utility. Source \( i \) attains a concave utility \( f_i(x_i) \) when it transmits at rate \( x_i \) that satisfies \( \underline{x}_i \leq x_i \leq \bar{x}_i \).

The NUM framework is general in that the choice of \( f_i \) allows for the representation of different goals of the net-
work operator. For example, using $f_i(x_i) = x_i$, maximizes throughput; setting $f_i(x_i) = \log(x_i)$ achieves proportional fairness among the sources; setting $f_i(x_i) = -1/x_i$ minimizes potential delay; these are common goals in communication network applications [7, 9].

Given the NUM formulation above, the algorithmic goal is to design a protocol that efficiently finds an approximately optimal solution. If the network is huge, it is often beneficial to distribute the solution, as performing the entire computation on a single machine is too costly [1, 12]. There is a large literature across the networked control and communication networks literatures that seeks to design such distributed optimization algorithms, e.g., [3]. We compare LOCO to the Alternating Method of Multipliers (ADMM) [1, 4].

In this work we focus on the throughput maximization case, i.e., $f_i(x_i) = x_i$; in this case NUM is an LP. The classical dual decomposition approach does not work for throughput maximization since it requires the objective function to be strictly concave [7]. However, ADMM can be applied.

Our complexity results hinge on the assumption that the constraint matrix $A$ is sparse. The sparsity of $A$ is defined as $\max(\alpha, \beta)$, where $\alpha$ and $\beta$ denote the maximum number of non-zero entries in a row and column of $A$ respectively. Formally, we say that $A$ is sparse if the sparsity of $A$ is bounded by a constant. This assumption usually holds in network control applications since $\alpha$ is the maximum number of sources sharing a link, which is typically small compared to $n$, and $\beta$ is the maximum number of links each source uses, which is typically small compared to $m$.

3. LOCAL CONVEX OPTIMIZATION

In this section, we introduce a local algorithm for distributed convex optimization, Local Convex Optimization (LOCO). In LOCO, every source in the network computes its portion of a near optimal solution using a small number of messages, without needing global communication or iteration. This is in contrast to iterative descent methods, e.g. ADMM, which are global, i.e., they spread the information necessary to find an optimal solution throughout the whole network over a series of rounds.

Distributed algorithms for NUM should perform well on two measures. The first is message complexity: the number of messages that are sent across links of the network in order to compute the solution. For both our algorithm and ADMM the message length will be of order $O(\log n)$. The second is the approximation ratio, which measures the quality of the solution provided by the algorithm. Specifically, an algorithm is said to $\alpha$-approximate a maximization problem if its solution is guaranteed to be at least $\frac{OPT}{\alpha}$, where $OPT$ is the value of the optimal solution. LOCO has provable worst-case guarantees on both its approximation ratio and message complexity, and improves on the communication overhead of iterative descent methods by orders of magnitude in practice when asked to compute a piece of the optimal solution.

3.1 An overview of LOCO

The key insight in the design and analysis of LOCO is that any natural online optimization algorithm can be converted into a local, distributed optimization algorithm. Note that the resulting distributed algorithm is for a static problem, not an online one. Further, after this conversion, the distributed optimization algorithm has the same approximation ratio as the original online optimization algorithm. Thus, given an optimization problem for which there exist effective online algorithms, these online algorithms can be converted into effective local, distributed algorithms.

More formally, to reduce a static optimization problem to an online optimization problem, we do the following. Let $Y$ be the set of constraints of an optimization problem $P$. Let $r : Y \rightarrow [0, 1]$ be a ranking function that assigns each constraint $y_j$ a real number between 0 and 1, uniformly at random. We call $r(y_j)$ $y_j$’s rank. Suppose that there is some online algorithm that receives the constraints sequentially and must augment the variables immediately and irrevocably so as to satisfy each arriving constraint. Suppose furthermore that for each constraint $y_j$, we can pinpoint a small set of constraints $S(y_j)$ (which we call $y_j$’s query set) that arrived before it so that restricting the set of constraints of $P$ to $S(y_j)$ results in the algorithm producing exactly the same solution for the variables that are present in $y_j$. Then simulating the algorithm on only $S(y_j)$ would suffice to obtain the solution for the variables in constraint $y_j$. More specifically, the steps of LOCO are as follows.

**Step 1**, Generating a localized neighborhood. For clarity, we break the first step into three sub-steps.

**Step 1a**, Representing the constraint matrix as a bipartite graph. A boolean matrix $A$ can be represented as a bipartite graph $G = (L, R, E')$ as follows. Each row of $A$ is represented by a vertex $v_l \in L$; each column by a vertex $v_r \in R$. The edge $(v_l, v_r)$ is in $E'$ if and only if $A_{l,r} = 1$. A more intuitive way to interpret $G$ is the following: $L$ represents the variables, $R$ the constraints. Edges represent which variables appear in which constraints. Note that the maximum degree of $G$ is exactly the sparsity of $A$.

**Step 1b**, Constructing the dependency graph. We construct the dependency graph $H = (V, E)$ as follows. The vertices of the dependency graph are the vertices of $R$; an edge exists between two vertices in $H$ if the corresponding vertices in $G$ share a neighbor. Intuitively, $H$ represents the “direct dependencies” between the constraints: changing the value of any variable immediately affects all constraints in which it appears, hence these constraints can be thought of as directly dependent. The maximum degree of $H$ is upper bounded by the square of the sparsity of $A$.

**Step 1c**, Constructing the query set. In order to build the query set, we generate a random ranking function on the vertices of $H$, $r : V \rightarrow [0, 1]$. Given the dependency graph $H$, an initial node $y \in V$ and the ranking function $r$, we build the query set of $y$, denoted $S(y)$, using a variation of BFS, as follows. Initialize $S(y)$ to contain $y$. For every vertex $v \in S(y)$, scan $v$’s neighbors, denoted $N(v)$, and add them to $S(y)$. Continue iteratively until no more vertices can be added to $S(y)$ (that is, for every vertex $v \in S(y)$ all of its neighbors that are not themselves in $S(y)$ have lower rank than $v$). If there are ties (i.e., two neighbors $u, v$ such that $r(u) = r(v)$), tie-break by ID.

**Step 2**, Simulating the online algorithm. Assume that we have an online algorithm for the problem that we would like LOCO to solve. In this paper we use the online packing
Algorithm of Buchbinder and Naor [2, chapter 14]. We provide the pseudocode in the extended version of the paper [6], for completeness. The specific setting that the online algorithm must apply to is the following: the variables of the convex program are known in advance, as are the univariate constraints. The rest of the constraints arrive one at a time; the online algorithm is expected to satisfy each constraint as it arrives, by increasing the value of some of the variables. It is never allowed to decrease the value of any variable. We simulate the online algorithm as follows:

In order to compute its value in the solution, source $i$ applies $r$ to the set of constraints in which it is contained, $Y(i)$. For $y = \arg\max_{z \in Y(i)} \{r(z)\}$, it simulates the online algorithm on $S(y)$. That is, it executes the online algorithm on the neighborhood constructed in Step 1 for the "last arriving" constraint that contains $x_i$. The constraints arrive in the order defined by $r$. The resulting $x_i$ value is identical to its value if the online algorithm was executed on the entire program, with the constraints arriving in the order defined by $r$.

### 3.2 Performance of LOCO

Our main theoretical result shows that LOCO can compute solutions to convex optimization problems that are as good as those of the best online algorithms for the problems, while using very little communication. We then specialize this case to throughput maximization in NUM. While we focus on NUM in this paper, the theorem and its proof apply to a wider family of problems as well. Specifically, the conversion from online to local outlined below can be used more broadly for any class of optimization problems for which effective online algorithms exist.

**Theorem 1.** Let $P$ be a problem with a concave objective function and linear inequality constraints, with $n$ variables and $m$ constraints, whose constraint matrix has sparsity $\sigma$. Given an online algorithm for $P$ with competitive ratio $h(n, m)$, there exists a local computation algorithm for $P$ with approximation ratio $h(n, m)$ that uses $2^{O(\sigma^2)} \log(n + m)$ messages with probability $1 - 1/poly(n, m)$.

We also have the following result for NUM with a linear objective function, which corresponds to maximizing throughput in NUM.

**Theorem 2.** Let $P$ be a throughput maximization problem with $n$ variables, $m$ constraints, and a sparse constraint matrix. LOCO computes an $O(\log m) - \epsilon$ approximation to the optimal solution of $P$ using $O(\log(n + m))$ messages with probability $1 - 1/poly(n, m)$.

In a simulation study, we focus on the case of maximizing throughput, demonstrating the empirical performance of LOCO on both synthetic and real networks. The results highlight that an orders-of-magnitude reduction in communication is possible with LOCO as compared to ADMM. For a description of our experimental setup, please see the full version [6].

Figure 1(a) highlights that LOCO requires considerably fewer messages than ADMM, across both small and large $n$. Both the average and maximum amount of communication needed to answer a query about a specific piece of the solution under LOCO (LOCO Avg and LOCO Max respectively) are substantially lower than for ADMM. Even answering every query (LOCO Tot) requires only the same order of magnitude as ADMM. The figure includes ADMM with a tolerance $\epsilon/10$ of $10^{-4}$ (ADMM 1) and $10^{-3}$ (ADMM 2) [1]. Even with suboptimal tolerance, which results in fewer iterations, ADMM still requires orders of magnitude more communication than LOCO.

Figure 1(b) illustrates the Pareto optimal frontier for ADMM: the minimal messages needed in order to obtain a particular relative error. Interestingly, the total number of messages used by LOCO is only slightly outside the Pareto frontier of ADMM, indicating that the local advantages that LOCO offers do not incur much global overhead.

### 4. REFERENCES


Electric vehicle charging: a queueing approach

Angelos Aveklouris
Eindhoven University of Technology
a.aveklouris@tue.nl

Yorie Nakahira
California Institute of Technology
ynakahir@caltech.edu

Maria Vlasiou
Eindhoven University of Technology
m.vlasiou@tue.nl

Bert Zwart
Eindhoven University of Technology,
Centrum Wiskunde en Informatica
bert.zwart@cwi.nl

ABSTRACT
The number of electric vehicles (EVs) is expected to increase. As a consequence, more EVs will need charging, potentially causing not only congestion at charging stations, but also in the distribution grid. Our goal is to illustrate how this gives rise to resource allocation and performance problems that are of interest to the Sigmetrics community.

1. INTRODUCTION
EVs consume a large amount of energy and as a result the charging of EVs is causing congestion in the distribution grid [3], which is exacerbated as the number of charging stations is limited. Motivated by this, we consider a stylized model that models the interplay of two sources of congestion (as not all cars find a space): (i) the number of available spaces with charging stations; (ii) the amount of available power.

Despite being a relatively new topic, the engineering literature on EV charging is huge. Here, we give only a sample. In [15], an algorithm for optimally managing a large number of plug-in EVs charging at a parking station is suggested. In [14], optimal charging algorithms that minimize the impact of plug-in EV charging on the connected distribution grid are proposed. Examples of studies where randomness is taken into account are [10], in which a methodology of modeling the overall charging demand of plug-in EVs is proposed, and [16] where control algorithms based on randomized EV arrival time are suggested. Mathematical models where vehicles communicate beforehand with the grid to convey information about their charging status are studied in [13]. In [8], cars are the central object and a dynamic program is formulated that prescribes how cars should charge their battery using price signals. Though the class of problem at hand fits well to the performance analysis, the only other line of work where such ideas are used is [1] and [17], where a gradient scheduler is proposed to minimize delays.

A common feature of the above studies is that they apply to shorter operational time-scales. Since the desired scale of infrastructure does not exist yet, it is important to consider models that can be used on longer (investment) time-scales. Equilibrium models are quite popular for investment and policy analysis of energy systems [4]. We therefore consider a stylized equilibrium queueing model that takes into account both congestion in the distribution grid, as well as congestion in the number of available spaces with charging stations. We consider a stylized model of a parking lot with finitely many spaces in which EVs (customers) arrive randomly in order to get charged (for another application of queueing theory to parking lots see [9]). The EVs have a random parking time and a random energy demand. Thus, each EV receives two kinds of service, parking and charging. We assume that all available power is charged at the same rate to all cars that need charging; some of our results can be extended to time-varying arrival rates and multiple types of users and stations, but due to space we do not do so here.

Under Markovian assumptions, our analysis focuses on the probability that an EV leaves the parking lot with a fully charged battery. Specifically, we develop bounds and a fluid approximation, and report partial results on a diffusion approximation. Our mathematical results are closely related to work on processor-sharing queues with impatience [7], though the model here is more complicated as there is limited number of spaces in the system and fully charged cars may not leave immediately as they are still parked.

2. MODEL DESCRIPTION
We consider a charging station with \( K > 0 \) parking spaces, each having an EV charger. We assume that the arrival, parking and charging times of EVs are mutually independent, and exponential with rates \( \lambda, \mu \) and \( \nu \), respectively; EVs leaves the system after their parking time expires. An EV may leave the system without its battery being fully charged. Furthermore, if all spaces are occupied, a newly arriving EV does not enter the system but leaves immediately. As such, the total number of vehicles in the system can be modeled by an Erlang loss system, though we need a more detailed description of the state space.

We denote by \( Q(t) \in \{0, 1, \ldots, K\} \) the total number of EVs in the system at time \( t \geq 0 \), where \( Q(0) \) is the initial number of EVs. Further, we denote by \( U(t) \in \{0, 1, \ldots, Q(t)\} \) the number of EVs of which their battery is not fully charged at time \( t \) and by \( U(0) \) the number of vehicles initially in the system. Thus, \( C(t) = Q(t) - U(t) \) represents the number of EVs with a fully charged battery at time \( t \).
The power consumed by the parking lot is limited and depends on the number of uncharged EVs at time $t$. We let it be given by $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$, $f(U(t)) := \min(U(t), M)$. Here, $0 < M \leq K$ denotes the maximum number of cars the parking lot can charge at full power.

3. MAIN RESULTS

We present bounds and approximations based on fluid and diffusion limits for the fraction of EVs that get fully charged. Proofs (and results for other performance measures) will be presented in an extended version of this paper.

3.1 Bounds

Under our assumptions, the number of uncharged EVs and the total number of EVs in the system $(U(t), Q(t))$, is a two dimensional Markov process. The fraction of fully charged EVs in stationarity is given by the ratio $\frac{\mathbb{E}[Q(t)]}{\mathbb{E}[U(t)]}$. In the special case $K = M$, we can compute explicitly the joint distribution, and in the case $K = \infty$, the distribution of the number of uncharged EVs is given by a variation of the Erlang A formula (see [18] for details on the Erlang A model). Note that, in our model customers / EVs can leave the system also during their service, unlike in the Erlang A queue. Based on these two special cases ($K = M$, $K = \infty$), the following proposition, which can be proved using Markov-rewards methods, presents an upper and a lower bound for the fraction of EVs that get fully charged.

**Proposition 3.1.** Let $C_{Q,n}^N(\infty)$ and $Q_{Q,n}^N(\infty)$ be the number of fully charged EVs and the total number of EVs in stationarity for the system $(K, M)$. We have that

$$\mathbb{E}[C_{Q,n}^N(\infty)] \leq \mathbb{E}[C_{Q,K}^N(\infty)] \leq \mathbb{E}[C_{Q,M}^N(\infty)].$$

3.2 Fluid approximation

We develop a fluid approximation for finite $K$, following a similar approach as in [7]. The main differences are the finitely many servers in the system and that the state space consists of two regions: $U(\infty) > M$ and $U(\infty) \leq M$.

Consider a family of models as defined earlier indexed by $n$. The fluid scaling (in steady state) is given by $\frac{U_n}{M}(t)$. To obtain a non-trivial fluid limit, we assume that the capacity of power in the $n$th system is given by $nM$, the arrival rate by $n\lambda$, and the number of parking spaces by $nK$.

**Proposition 3.2.** Let $E_{\mu}$ and $E_{\nu}$ be exponential random variables with rates $\mu$ and $\nu$. We have that $U_n(t) \rightarrow u^*$, as $n \rightarrow \infty$. In addition, $u^*$ is given by the unique positive solution of the following fixed-point equation:

$$u^* = \min\{\lambda, \mu K\} \mathbb{E}[\min\{E_{\mu}, E_{\nu} \max\{1, u^* / M\} \}].$$

Observe that if we define $f(U(\cdot)) = 1$ (i.e., the processor sharing discipline) and replace $K$ by $n\lambda K$ (assuming for simplicity $\mu = 1$), we derive [7, Equation 4.1].

We directly use a modified form of our fluid approximation, which can be derived heuristically using Little’s law and a version of the snapshot principle (essentially assuming an EV sees the system in stationarity throughout its sojourn). Let $P_K$ be the blocking probability in a loss system with $K$ servers. To obtain our approximation, we replace $\min\{\lambda, \mu K\}$ by $\lambda(1 - P_K)$, leading to

$$u^* = \lambda(1 - P_K) \mathbb{E}[\min\{E_{\mu}, E_{\nu} \max\{1, u^* / M\} \}].$$

Let $P_s$ denote the probability that an EV leaves the parking lot with fully charged battery in the fluid model. It is given by $P_s = \mathbb{P}(E_{\mu} > E_{\nu} \max\{1, u^* / M\})$, where $u^*$ is the unique solution of (2). Under our assumptions, the explicit expression for this probability can be found. That is,

$$P_s = \begin{cases} \frac{\nu}{\nu + \mu}, & u^* \leq M, \\
\frac{\nu}{\nu + \mu}, & u^* > M. \end{cases}$$

3.3 Diffusion Approximation

Let $\beta$ and $\kappa$ be real numbers. Consider the asymptotic regime. Define $M_n = \frac{\lambda_n}{\mu_x} + \beta / \sqrt{n}$ and $\lambda_n = n(\nu + \mu_d)$, i.e., the “square-root staffing rule” as in [5] and [6]. In addition, define $K_n = \frac{\lambda_n}{\mu_y} + \kappa / \sqrt{n}$. The diffusion scaling is given by $\hat{U}_n(t) := \frac{U_n(t) - \lambda_n}{\mu_x} M$ and $\hat{Q}_n(t) := \frac{Q_n(t) - \mu_x}{\mu_x} M$.

**Theorem 3.3.** If $(\hat{U}_n(0), \hat{Q}_n(0)) \xrightarrow{d} (U(0), Q(0))$ then $(\hat{U}_n(\cdot), \hat{Q}_n(\cdot)) \xrightarrow{d} (\hat{U}(\cdot), \hat{Q}(\cdot))$, as $n \rightarrow \infty$. The diffusion limit satisfies the following 2-dimensional stochastic differential equation

$$\begin{bmatrix}
d\hat{U}(t) \\
d\hat{Q}(t)
\end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2(\hat{\nu} + \mu)} - \beta}{\sqrt{\nu + \mu}} & 0 \\
\frac{\beta}{\sqrt{\nu + \mu}} & \frac{1}{\nu + \mu}
\end{bmatrix} \begin{bmatrix} d\hat{W}_1(t) \\
d\hat{W}_2(t)
\end{bmatrix} + \begin{bmatrix} b_1(\hat{U}(t), \hat{Q}(t)) \\
b_2(\hat{U}(t), \hat{Q}(t))
\end{bmatrix} dt - \begin{bmatrix} d\hat{Y}(t) \\
d\hat{Y}(t)
\end{bmatrix},$$

where $b_1(x, y) = -\beta(x \wedge \beta) - \mu_x$ and $b_2(x, y) = -\mu_y$. Further, $\hat{W}_1(t)$ and $\hat{W}_2(t)$ are driftless, univariate Brownian motions such that $2(\hat{\nu} + \mu)\mathbb{E}[dW_1(t)]d\hat{W}_2(t) = (\nu + 2\mu)dt$. In addition, $\hat{Y}(\cdot)$ is the unique nondecreasing nonnegative process such that (4) holds and $f_0 \mathbb{E}\left[\int_{U(\cdot) < \infty} d\hat{Y}(t)\right] = 0$.

Note that $Q(t)$ satisfies the known Erlang B diffusion [11]. When $\kappa = \infty$ the system (4) has an explicit invariant distribution. Take the vectors $m_\cdot = (0, 0)$, $m_+ = (-\frac{\nu}{\nu + \mu}, 0)$ and the positive definite matrices $\Sigma_- = \begin{bmatrix} 1 & \frac{1}{\nu + \mu} \\
\frac{1}{\nu + \mu} & \frac{1}{\nu + \mu}
\end{bmatrix}$ and $\Sigma_+ = \begin{bmatrix} \frac{\nu + \mu}{\nu + \mu} & \frac{1}{\nu + \mu} \\
\frac{1}{\nu + \mu} & \frac{1}{\nu + \mu}
\end{bmatrix}$. Let $f_-$ and $f_+$ be 2-dimensional normal pdfs with mean vectors $m_-$, $m_+$ and covariance matrices $\Sigma_-$, $\Sigma_+$, respectively. In case $K = \infty$, we can show that the joint steady state pdf of the random vector $(\hat{U}(\infty), Q(\infty))$ can be written as

$$\phi(x, y) = c_1 f_-(x, y) 1_{x \in \mathbb{S}_1} + c_2 f_+(x, y) 1_{x \in \mathbb{S}_2},$$

where $c_1, c_2$ are given in [5, Equations 3.9–3.10].

3.4 Numerical evaluation and discussion

In Fig. 1–3 we depict the bounds in (1) and the fluid approximation in (3) for 3 cases: moderately, critically, and over-loaded. Further, we fix $\nu = \mu = 1$. The vertical axes give the probability that an EV leaves the parking lot with fully charged battery (success probability) and the horizontal axes give the ratio $M/K$. The lower and the upper bounds seem to be tight for $M > 0.7K$. Also, the lower bound is tight under light load, in the other cases the fluid approximation works well for $K = 50$. 

REFERENCES

5. REFERENCES

4. DISCUSSION AND EXTENSIONS

Our numerical results show there is room for improvement for critically loaded systems, making it worthwhile to derive the invariant distribution of the process in Theorem 3.4; the solution for \( \kappa = \infty \) did not yield better results than the Erlang A lower bound.

From an applications standpoint, it is important to remove various model assumptions. If parking and charging times are given by the (possibly dependent) generally distributed random variables \( B \) and \( D \), we can develop a measure-valued fluid model by extending [7]. The fluid limit in steady-state will be defined by the fixed point equation

\[
u^* = \lambda(1 - P_K)\mathbb{E}[\min\{B, D \max\{1, \frac{u^*}{M}\})].\]

We are currently extending this to time-varying arrival rates, multiple customer classes, and multiple parking lots. The distribution network for the latter is modeled as a tree network in [2] where simulation results are presented. On a high level, the analysis is reminiscent of [12].

5. REFERENCES


Optimally Scheduling Jobs with Multiple Tasks

Ziv Scully; Guy Blelloch, Mor Harchol-Balter†
Computer Science Department
Carnegie Mellon University
{zscully, blelloch, harchol}@cs.cmu.edu

Alan Scheller-Wolf
Tepper School of Business
Carnegie Mellon University
awolf@andrew.cmu.edu

ABSTRACT
We consider optimal job scheduling where each job consists of multiple tasks, each of unknown duration, with precedence constraints between tasks. A job is not considered complete until all of its tasks are complete. Traditional heuristics, such as favoring the job of shortest expected remaining processing time, are suboptimal in this setting. Furthermore, even if we know which job to run, it is not obvious which task within that job to serve. In this paper, we characterize the optimal policy for a class of such scheduling problems and show that the policy is simple to compute.

1. INTRODUCTION
Scheduling jobs of unknown service requirements in preemptive multiclass single-server queueing systems is a classic, well-studied problem. The optimal choice of scheduling policy for minimizing mean response time depends on the job size distributions. For instance, the shortest expected remaining processing time (SERPT) policy is optimal for some size distributions, but other policies, such as highest hazard rate (HHR), are optimal for others [1]. In this setting, researchers usually treat each job as a single chunk of work that must be served for a certain length of time until completion.

In this paper, we introduce and study a new single-processor scheduling problem in which jobs consist of multiple tasks. Our job model differs from standard models in several ways.

• Jobs may have multiple tasks. A task is a single preemptible chunk of work that must be served for a certain length of time until it completes. The processor serves individual tasks rather than jobs as a whole. Tasks have unknown sizes drawn from known distributions.
• Tasks within each job are subject to precedence constraints. The constraints keep the scheduler from serving certain tasks until others have completed. There are never constraints between tasks of different jobs.
• A job exits the system when all of its tasks are complete. We measure response time of jobs, so there is no “partial credit” for completing only some of a job’s tasks.

The central question is: what scheduling policy minimizes mean response time? Specifically, at every moment in time, which job should we run, and which task of that job should we serve? In fact, we permit sharing the processor between multiple tasks of multiple jobs, further complicating matters.

We call this problem single-processor multitask scheduling, or simply multitask scheduling, and we call the jobs multitask jobs. In this paper, we provide the first analysis of multitask scheduling, deriving a provably optimal policy for a case in which all jobs are present at the start. It is worth highlighting with some examples how multitask scheduling differs from standard scheduling problems and what makes it challenging.

Example 1.1. Suppose job $J$ has a single task of size $S_1 + S_2$ and job $K$ has two tasks, one each of size $S_1$ and $S_2$, where $S_1$ and $S_2$ are arbitrary size distributions. Suppose further that a precedence constraint forces completing $K$’s task of size $S_1$ first. Although both $J$ and $K$ have total size $S_1 + S_2$, they are very different: in a scenario in which $J$ and $K$ have received service for the same amount of time, we have more information about $K$’s remaining size than $J$’s because we know whether $K$’s first task is complete. We can use this information when scheduling to reduce mean response time.

Example 1.2. Suppose jobs $J$ and $K$ each have 5 tasks with no precedence constraints. Each of $J$’s tasks takes either 1 second, with probability 0.9, or 100 seconds, with probability 0.1. Each of $K$’s tasks takes 10 seconds. In expectation, $J$ is longer than $K$, so SERPT (using total size) would run $K$ first. However, with probability $0.9^5 \approx 0.59$, all of $J$’s tasks are short, in which case we would rather run $J$ first. It turns out the optimal policy first serves tasks from $J$ but switches to $K$ if any of $J$’s tasks turns out to be long.

Example 1.3. Suppose jobs $J$ and $K$ have 2 and 3 tasks, respectively, with no precedence constraints. All tasks have the same Pareto size distribution, but they have received different amounts of service previously. $J$’s tasks have ages $j_1$, and $K$’s tasks have ages $k_i$ with $j_1 > j_2 > k_1 > k_2 > k_3 > m$, where $m$ is the minimum possible value of the Pareto distribution. Should we run $J$ or $K$ first? Because Pareto distributions are heavy-tailed, $K$’s tasks are shorter in expectation, but $J$ has fewer tasks. SERPT is a natural heuristic, but it can make the wrong tradeoff. HHR is optimal for single-task jobs with Pareto size distributions [1], but it is not clear how to define hazard rate with multiple tasks. Once we choose a job to run, we still have to choose a task to serve, and unlike the previous example, there is no clear “short vs. long” intuition.

In summary, none of the common heuristics work. To solve this case, we need new machinery. We show in Section 5 how to compute the optimal policy for this example. Curiously, within each job, it is optimal to serve the longest task first.

−Supported by an ARCS Foundation scholarship and the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1252522. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

†This research was supported by NSF-XPS-1629444, NSF-CMMI-1538204, NSF-CMMI-1334194, and Faculty Awards from Google and Facebook.

Copyright is held by author/owner(s).
**Contributions.** In this paper, we give the first theoretical analysis of multitask scheduling. Our contribution is two-fold: we give an optimal policy for a case of multitask scheduling, and, in deriving the policy, we introduce two novel techniques.

We derive the optimal policy for multitask scheduling with jobs whose tasks have “aged Pareto” size distributions, as in Example 1.3 (see Section 5). The policy is distinct from SERPT, HHR, and other common heuristics. Though it has no closed-form description, it is easy to compute numerically.

Our approach is built on the theory of the Gittins index. Though the Gittins index has been applied to scheduling problems in the past [1], there are major obstacles to applying it to multitask scheduling (see Section 2). We introduce a pair of new tools that help overcome these obstacles. The composition law simplifies Gittins index computations for jobs whose tasks can be grouped into “phases” that must be served in a fixed order (see Section 3). The autopiloting law adds “imaginary precedence constraints” to certain jobs (see Section 4). This is crucial for proving optimality of the Gittins index approach for such jobs, and it makes the optimal policy practical to compute using the above composition law.

There has been much progress in the study of scheduling monolithic single-task jobs. However, real-world applications in domains such as web services [2], pharmaceuticals [5], and data analytics [8] increasingly deal with jobs consisting of multiple tasks. Our long-term goal is to understand scheduling of such jobs in multiple-processor, heterogeneous-server, and other realistic settings. This paper presents an initial step towards this goal in analyzing single-processor multitask scheduling, which already presents significant challenges.

**Related Work.** Task graph scheduling bears a resemblance to multitask scheduling. Roughly speaking, task graph scheduling considers scheduling a single multitask job, usually with known task sizes, on multiple processors. In contrast, single-processor multitask scheduling considers scheduling multiple multitask jobs with unknown task sizes. Task graph scheduling is NP-complete [6], but simple heuristics give constant-factor approximations.

Another related problem is scheduling jobs with interjob precedence constraints [3, Section 4.6]. The problem appears similar to multitask scheduling and has also been studied using the Gittins index, but it differs in two important ways: the precedence constraints are between separate jobs, whereas our precedence constraints are between tasks within a single job, and the solution treats the nonpreemptive case, whereas preemption is what makes multitask scheduling hard.

## 2. GITTINS INDEX BACKGROUND

The Gittins index was originally introduced by Gittins and Jones [4] to solve the multi-armed bandit problem. In its 45-year history, it has been applied to a menagerie of optimization problems [3]. The Gittins index theorem [3, Section 3.3] is a result for standard single-task scheduling which shows that the Gittins index policy, which we call Gittins scheduling, always minimizes mean response time. Gittins scheduling always serves the (single-task) job of maximal Gittins index, which is a quantity computed for each job independently of other jobs in the system. The Gittins index theorem can also apply to multitask jobs, but only under certain conditions (see Definition 2.3).

There are several equivalent definitions of the Gittins index. Our definition uses an unorthodox auxiliary optimization problem but is easily shown to agree with usual definitions.

Single-job profit (SJP) is an optimization problem concerning a multitask job \( J \) and a potential reward \( r \geq 0 \). At every instant, we choose between running the job and giving up. The process ends when the job completes or we choose to give up, whichever comes first. If the job completes, we receive value \( r \), but we are continuously charged value at rate 1 while serving the job. The goal is to maximize expected net value.

We call a policy for SJP a *job policy*. A job policy \( \pi = (\sigma, \tau) \) has two components: a *stopping policy* \( \sigma \), deciding when to run the job and when to give up, and a *task policy* \( \tau \), deciding which task of the job to serve while running it.

**Definition 2.1.** In SJP with job \( J \) and reward \( r \), the utility of job policy \( \pi \) is

\[
U[r](J, \pi) = rP(\pi \text{ completes } J) - E[\text{time } \pi \text{ runs } J],
\]

and the optimal profit, or simply profit, is

\[
V[r](J) = \sup_{\pi} U[r](J, \pi),
\]

where \( \pi \) ranges over job policies. The profit function of job \( J \) is the profit as a function of reward, \( V[\cdot](J) \).

SJP can measure how “desirable” it is to run a job. If we increase the reward starting from 0, the profit also increases starting from 0 and eventually becomes positive. The point at which profit becomes positive is the minimum reward that entices us to run the job for at least an instant in SJP. Jobs that require larger rewards are less desirable to run.

**Definition 2.2.** In SJP with job \( J \), the fair reward is

\[
R(J) = \inf \{ r > 0 \mid V[r](J) > 0 \},
\]

and the Gittins index of \( J \) is \( G(J) = 1/R(J) \).

Unfortunately, simple counterexamples exist showing that Gittins scheduling is not optimal in general for multitask scheduling. There is, however, a sufficient condition given by Whittle [7] which, when satisfied by every job, enables the proof of the Gittins index theorem.

**Definition 2.3.** A job \( J \) satisfies the *Whittle condition*, or is simply called *Whittle*, if there exists a task policy \( \tau^* \) such that for any reward \( r \),

\[
V[r](J) = \sup_{\pi(\sigma, \tau)} U[r](J, \pi) = \sup_{\pi(\sigma, \tau)} U[r](J, (\sigma, \tau^*)),
\]

where \( \sigma \) ranges over stopping policies and \( \tau \) ranges over task policies. We call such a task policy \( \tau^* \) an *autopilot* of \( J \).

The idea behind the Whittle condition is as follows. In SJP with job \( J \), the optimal job policy \( \pi = (\sigma, \tau) \) depends on the reward \( r \). Job \( J \) is Whittle if we can find a fixed task policy \( \tau^* \) that we can always use in our optimal job policy, regardless of \( r \). An optimal job policy for \( J \) is thus specified by just a stopping policy \( \sigma \), as we can optimally use \( \tau = \tau^* \).

In summary, a job is Whittle if we can pretend that we must always serve its tasks according to the autopilot. Roughly speaking, the fact that only the stopping policy varies allows the proof of the Gittins index theorem to go through [7].

**Theorem 2.4** (Gittins Index Theorem for Whittle Jobs). Gittins scheduling minimizes mean response time in multitask scheduling when all jobs are Whittle, provided that within each job, tasks are served according to the job’s autopilot.
Why Multitask Scheduling is an Open Problem.

We have seen that to solve multitask scheduling using the Gittins index, it suffices to (i) prove that all the jobs are Whittle and (ii) compute the Gittins indices of all the jobs. Both of these are very hard in general: the space of job policies for a job may be extremely large, which can make both the Whittle condition proof and Gittins index computation intractable. While there is prior work on the Whittle condition [3, Chapter 4], results are limited and do not directly apply to multitask scheduling. In the rest of this paper, we present two new techniques for overcoming these obstacles.

3. A NEW COMPOSITION LAW

One obstacle to using the Gittins index for multitask scheduling is that computing Gittins indices of general multitask jobs is very complicated. Our next result simplifies the Gittins index computation for jobs whose tasks can be grouped into “phases” that must be served in a fixed order.

Definition 3.1. The sequential composition of jobs \( J \) and \( K \), written \([J;K]\) is the multitask job consisting of the tasks from both \( J \) and \( K \) with their original precedence constraints. Additionally, all tasks from \( J \) have precedence over all tasks from \( K \). We call \( J \) and \( K \) the phases of \([J;K]\).

Theorem 3.2 (Composition Law). The profit function of a sequential composition is the composition of the phases’ profit functions. That is, for any jobs \( J \) and \( K \) and any reward \( r \),

\[
V[r]([J;K]) = V[V[r](K)](J).
\]

Proof. Because SJP is a Markov decision process, any SJP policy for \([J;K]\) decomposes into two policies: \( \pi_J \), operating during phase \( J \), and \( \pi_K \), operating during phase \( K \). Let

\[
P_J = E[\{\pi_J \text{ completes phase } J\}]
\]

\[
P_K = E[\{\pi_K \text{ completes phase } K | \pi_J \text{ completes phase } J\}]
\]

\[
E_K = E[\{\pi_K \text{ runs phase } K | \pi_J \text{ completes phase } J\}]
\]

Observe that running phase \( K \) of \([J;K]\) is identical to running just job \( K \). Recalling Definition 2.1, we compute

\[
V[r]([J;K]) = \sup_{\pi_J, \pi_K} (rP_J P_K - (E_J + P_J E_K))
\]

\[
= \sup_{\pi_J, \pi_K} (rP_J P_K - P_J - E_K) = V[V[r](K)](J).
\]

By Definition 2.2, computing the Gittins index of a job requires finding the largest zero of its profit function. Profit functions are increasing, convex, and Lipschitz continuous, so finding this zero is simple with numerical methods. Thus, to compute the Gittins index of a sequential composition, it suffices to compute the profit functions of its phases, which are much smaller individual computations.

4. A NEW AUTOPILOTING LAW

Another obstacle to using the Gittins index in multitask scheduling is that Gittins scheduling is only optimal when all the jobs are Whittle, and proving the Whittle condition can be very difficult in general. Our next result establishes a class of Whittle jobs.

The aged Pareto distribution of shape \( \alpha > 1 \) and age \( k > 0 \) is the distribution with tail \( F(t) = k^\alpha (k + t)^{-\alpha} \). It is the distribution of \( X \mid X > k \) for Pareto-distributed \( X \), provided \( k \) is at least \( X \)'s minimum possible value. Example 1.3 uses tasks of aged Pareto size. A job is multi-Pareto of shape \( \alpha \) if its tasks have aged Pareto sizes of the same shape \( \alpha \).

5. APPLYING THE LAWS

The composition law and autopiloting law combine to give the optimal policy for multitask scheduling of multi-Pareto jobs. The autopiloting law implies that Gittins scheduling is indeed the optimal policy, and we use both laws together to compute the Gittins index of a multi-Pareto job.

By the autopiloting law, we know it is always optimal to serve a multi-Pareto job’s tasks in a specific order, so we may imagine adding precedence constraints enforcing this order. We can then view the job as a sequential composition of single-task phases. Each phase is a single-task job \( J_{a,k} \) with aged Pareto size distribution of shape \( \alpha \) and age \( k \). A simple derivation gives the profit function of each phase:

\[
V[r](J_{a,k}) = \begin{cases} 
0 & \text{if } k < a \\
\frac{r - \frac{k}{\alpha(a-1)}(\alpha - (\frac{ak}{k})^{-(\alpha-1)})}{0} & \text{if } k > a
\end{cases}
\]

Thus, using the composition law, we can compute the job’s profit function, from which we can compute its Gittins index.

6. CONCLUSION AND FUTURE WORK

We propose a new problem, single-processor multitask scheduling, and solve it for a case where tasks have aged Pareto size distributions. To do so, we introduce two novel techniques, the composition and autopiloting laws, which greatly simplify the computation of the Gittins index policy and verify its optimality. This work opens up a huge space of new problems on multitask scheduling, from analyzing more task size distributions to considering multiple processors.

References

Large Deviations for Increasing Subsequences of Permutations and a Concurrency Application

Yuli Baryshnikov  
Depts. of Mathematics and ECE  
University of Illinois at Urbana-Champaign  
Urbana, IL 61801  
ymb@illinois.edu

Abram Magner  
Coordinated Science Lab  
University of Illinois at Urbana-Champaign  
Urbana, IL 61801  
anmagner@illinois.edu

ABSTRACT
The study of concurrent processes with conflict points is connected with the geometry of increasing subsequences of permutations – a permutation encodes the transactions of two processes that conflict (i.e., must be executed serially), and a given increasing subsequence encodes one particular serialization of the executions of two processes. This motivates the study of random increasing subsequences of random permutations. Here, we give a large deviation principle which implies that such a subsequence never deviates too far from the identity permutation: a random serialization of two concurrent processes will not favor either process too much at any given time. We then give an efficient exact algorithm for uniform random sampling of an increasing subsequence from a given permutation.

Categories and Subject Descriptors

General Terms
Algorithms, Performance, Theory.

Keywords
Large deviations, permutations, increasing subsequences.

1. INTRODUCTION
It is well known in the theory of concurrency control that the conflict points of a pair of processes can be represented geometrically as points in $[0,1]^2$ [4]. Given such a representation, a serialization of the two processes – a sequential scheduling of the constituent transactions of each process – corresponds exactly to a homotopy class of increasing paths from $(0,0)$ to $(1,1)$ which avoids all conflict points.

This geometric picture, in turn, corresponds exactly to the following more combinatorial setting: the $n$ conflict points correspond to a partial order $\sigma$ on a set of size $n$, and a serialization corresponds to an increasing subsequence of $\sigma$.

That is, we have the following:

**Proposition 1.** The number of distinct serializations of a pair of processes is equal to the number of increasing subsequences in the coordinatewise-domination partial order on the set of conflict points.

It is natural to study the typical behavior of serializations of a typical pair of processes, where the conflict points are now distributed uniformly at random in the $[0,1]^2$. Now $\sigma$ is a permutation with probability 1, and the goal is to study random increasing subsequences of a random permutation.

Here, we establish upper and lower bounds on large deviation probabilities for random increasing subsequences of a random permutation. For a pair of concurrent processes, this has the desirable consequence that a randomly selected serialization (usually) does not favor one process or another too much over any given time interval, so that neither process is subject to starvation.

Finally, we give an efficient exact algorithm for uniformly sampling an increasing subsequence of a given permutation.

2. PRIOR RESULTS
Increasing subsequences of random permutations (and, more generally, in random partial orders [1]) have been extensively studied. E.g., the moments of the number of increasing subsequences of a random permutation were precisely studied in [3] (notably implying poor concentration). Additionally, the length of the longest increasing subsequence of a random permutation has been characterized: its expected value is $2\sqrt{n(1+o(1))}$, with tight concentration around this value [2]. To our knowledge, large deviations for geometric properties of randomly chosen increasing subsequences of a random permutation have not been studied.

3. MODEL AND MAIN RESULTS
Consider a permutation $\sigma \in S_n$, the symmetric group on $n$ letters. An increasing subsequence of $\sigma$ is a finite sequence $1 \leq t_1 < t_2 < \cdots < t_k \leq n$ (possibly of length $k = 0$ or 1) of positive integers such that for all $j \in [k] = \{1,\ldots,k\}$, we have $\sigma(t_j) < \sigma(t_{j+1})$. We denote by $\tau(\sigma)$ the number of increasing subsequences of $\sigma$.

Here, we are interested in $\sigma$ sampled uniformly at random from $S_n$. Let $T = (t_1,\ldots,t_k)$ be a random increasing subsequence of $\sigma$.

Our first main result precisely characterizes the probability that any element of $T$ is far from the diagonal (in the sense that $|\sigma(i) - i|$ is large):

**Theorem 1 (LDP for increasing subsequences).** Let $\sigma$ be a random permutation from $S_n$, and let $T$ be a random increasing subsequence of $\sigma$. Then we have, for any
\( \epsilon > 0, \)
\[
\Pr \left[ \max_{i \in T} |\sigma(i) - i| > \epsilon n \right] = \exp(-\Theta(\sqrt{n})), \tag{1}
\]
where the constants hidden in the \( \Theta(\cdot) \) may be explicitly bounded in terms of \( \epsilon \).

In terms the concurrency scenario described in the introduction, this implies that a randomly chosen serialization of a pair of processes with randomly selected conflict points typically avoids the phenomenon of starvation.

It is also of interest (e.g., for the sake of simulation studies) to be able to sample a uniformly random increasing subsequence of a given permutation. Our next result says that this can be done efficiently.

**Theorem 2** (Exact sampling algorithm). There is an algorithm for sampling a uniformly random increasing subsequence of a permutation from \( S_n \) in time \( \Theta(n^2) \), assuming that arithmetic operations take constant time.

### 4. Proof of Theorem 1

The proof is guided by the following insight: if any given element of a random increasing subsequence strays too far from the diagonal (say, \( i \) is such that \( \sigma(i) \) is much larger than \( i \)), then the typical number of indices \( j > i \) for which \( \sigma(j) > \sigma(i) \) is small; thus, the number of increasing subsequences containing \( i \) (and, hence, the probability that a random increasing subsequence contains such an \( i \)) cannot be very large. Presently, we make this intuition rigorous.

We explain in detail how to upper bound the probability in \( (1) \); the lower bound is similar. To do this, we union bound over all \( i \in [n] \):
\[
\Pr \left[ \max_{i \in T} |\sigma(i) - i| > \epsilon n \right] \leq \sum_{i=1}^{n} \Pr[i \in T \cap |\sigma(i) - i| > \epsilon n].
\]

For simplicity, we will only upper bound the probability
\[
\Pr[i \in T \cap \sigma(i) - i > \epsilon n], \tag{2}
\]
which gives an upper bound for the probability of the desired event by symmetry.

To calculate this probability, we sum over all permutations \( \pi \) satisfying \( \pi(i) - i > \epsilon n \) and over all increasing subsequences \( t \) of \( \pi \):
\[
\Pr[\sigma(i) - i > \epsilon n \cap i \in T] = \sum_{\pi : \sigma(i) - i > \epsilon n} \frac{1}{n!} \sum_{t} \Pr[\pi = \pi \cap T = t] \tau(\pi, i), (3)
\]
where we have defined \( \tau(\pi, i) \) to be the number of increasing subsequences of \( \pi \) that include the index \( i \).

Let \( A_i = A_i(\sigma, i) \) denote the event that \( \sigma(i) - i > \epsilon n \). Then we can express \( (3) \) as a conditional expectation:
\[
\Pr[\sigma(i) - i > \epsilon n \cap i \in T] = \Pr[A_i(\sigma, i)] \cdot \mathbb{E} \left[ \frac{\tau(\pi, i)}{\tau(\sigma)} \right] A_i(\sigma, i), \tag{4}
\]
To compute the remaining expectation, we note that \( \tau(\pi, i) \) can be expressed as a product as follows:
\[
\tau(\pi, i) = \beta(\pi, i, \sigma(i)) \cdot \alpha(\pi, i, \sigma(i)), \tag{5}
\]
where we define \( \beta(\pi, i, j) = 1 + \# \) of nonempty increasing subsequences of \( \pi \) ending before \( i \) and whose last element \( x \) satisfies \( \pi(x) < j \), and \( \alpha(\pi, i, j) = 1 + \# \) of nonempty increasing subsequences of \( \pi \) beginning after \( i \) and whose initial element \( x \) satisfies \( \pi(x) > j \).

Now, both \( \beta \) and \( \alpha \) are constrained by the number of indices \( x \) for which, say, \( x < i \) and \( \pi(x) < \sigma(i) \). Thus, we introduce the following quantities:
\[
L_{i,k} := \{x \in [n]: x > i, \sigma(x) > k\},
\]
\[
B_{i,k} := \{x \in [n]: x < i, \sigma(x) < k\}.
\]

Here, \( B_{i,k} \) is hypergeometrically distributed with population size \( n \), number of trials \( i - 1 \), and number of successes \( k - 1 \). Similarly, for any given \( b \), \( L_{i,k}[B_{i,k} = b] \) is hypergeometrically distributed with population size \( n - i \), number of trials \( n - i \), and number of successes \( n - k - i + b \).

With this in mind, we define an event \( W \), under which \( L_{i,\sigma(i)} \) and \( B_{i,\sigma(i)} \) are well-behaved: for an arbitrary fixed \( t > 0 \),
\[
W = \left[ |B_{i+t+i+\epsilon n} - \mathbb{E}[B_{i+t+i+\epsilon n}]| < t(i-1) \right. \\
\left. \cap |L_{i+t+i+\epsilon n} - \mathbb{E}[L_{i+t+i+\epsilon n}]| < t(n-i) \right].
\]

Then from standard hypergeometric concentration results,\n\[
\Pr[W] \geq 1 - e^{-2t^2\epsilon^2(1-t)} - e^{-2t^2\epsilon^2(n-t)}.\tag{6}
\]

We can rewrite the expectation in \( (4) \) by conditioning on \( W \):
\[
\mathbb{E}\left[ \frac{\tau(\pi, i)}{\tau(\sigma)} | A_i(\sigma, i) \right] \leq \mathbb{E}\left[ \frac{\tau(\pi, i)}{\tau(\sigma)} | A_i(\sigma, i) \right] + \Pr[\neg W | A_i(\sigma, i)],
\]
where the inequality is from the fact that \( \tau(\pi, i)/\tau(\sigma) \leq 1 \). The second term decays exponentially, by our choice of \( W \), so we can focus on the first term.

To proceed, we will need a lemma to the effect that we can split the expected fraction into a ratio involving two expectations that are easy to bound:

**Lemma 3.** We have
\[
\mathbb{E}\left[ \frac{\tau(\pi, i) | I(W) \cap A_i(\sigma, i)}{\tau(\sigma)} \right] \leq \frac{\mathbb{E}[\tau(\pi, i) | I(W) | A_i(\sigma, i)]}{2\mathbb{E}[L(\pi)]} + e^{-\Omega(n/\log^3 n)}. \tag{7}
\]

Here, \( L(\pi) \) denotes the length of the longest increasing subsequence of the \( \pi \), so the denominator becomes
\[
\mathbb{E}[L(\pi)] = 2\sqrt{\pi(1+\epsilon(n))}. \tag{7}
\]

We can then estimate the numerator of the ratio \( (6) \) as follows:
\[
\mathbb{E}[\tau(\pi, i) | I(W) | A_i(\sigma, i)] = \mathbb{E}[\beta(\sigma, i, \sigma(i)) \cdot \alpha(\sigma, i, \sigma(i)) | I(W) | A_i(\sigma, i)].
\]

We can evaluate the remaining expression by observing that, conditioned on particular values for \( B_{i+t+i+\epsilon n} \) and \( L_{i+t+i+\epsilon n} \), \( b \) and \( \epsilon \), which are constrained by \( T \), \( \beta(\pi, i, \sigma(i)) \) and \( \alpha(\pi, i, \sigma(i)) \) are independent and have the same distribution as the number of increasing subsequences in random permutations from \( S_b \) and \( S_{b+\epsilon} \), respectively (the first and second moment of this distribution are given in \( [3] \)).

This implies that
\[
\mathbb{E}[\tau(\pi, i) | I(W) | A_i(\sigma, i)] \leq 2\sqrt{\frac{(1+\epsilon(n))}{n(1+O(1/n))} + 2\sqrt{n - (i+\epsilon(n)) - i+t + t(i+n)\epsilon(n)}}.
\]
Now, suppose that $i \sim cn$, for $c \in (0, 1)$. Then the expression in the exponent becomes $2\sqrt{\pi} \sqrt{c(i + \epsilon) + \sqrt{1 - 2c - \epsilon + c(c + \epsilon)}}$, and we want to choose $\epsilon$ so that this is $< 2\sqrt{\pi}$. We have the following:

$$\sqrt{c(i + \epsilon) + \sqrt{1 - 2c - \epsilon + c(c + \epsilon)}} < \sqrt{c(1 + \epsilon/c) + (1 - c)(1 - \epsilon/(1 - c))} = 1,$$

where the inequality is by Jensen and the concavity of the square root function, and it is strict because $\epsilon \in (0, 1) - c$.

Now, note that the function

$$F(c) = \sqrt{c^2 + \sqrt{1 - 2c - \epsilon + c(c + \epsilon)}}$$

satisfies $F(0) = 1 - \epsilon$ and $F(1 - \epsilon) = \sqrt{1 - \epsilon}$ and is concave as a function of $c$.

This implies that any nonzero value of $\epsilon$ yields exponential decay of the probability $\Pr[\sigma(i) - i > cn \cap i \in T]$; i.e., it is at most $e^{-\Theta(\sqrt{n})}$, where the constant in the $\Theta(\cdot)$ is bounded away from 0 (for any fixed $\epsilon$) for all $i$.

Now, to upper bound $\Pr[\sigma(i) - i > cn \cap i \in T]$, we note the following symmetry property: an increasing subsequence of $\sigma$ containing $i$ is equivalent to an increasing subsequence of the counterclockwise rotation of the permutation $\sigma$ corresponding to the graph of $\sigma$ by $\pi$ radians containing $n - i$. Moreover, $i - \sigma(i) > cn$ if and only if $\rho(n - i) - (n - i) > cn$, and $\rho$ is uniformly distributed on $S_n$. This implies that

$$\sum_{i=1}^{n} \Pr[i \in T \cap \sigma(i) - i > cn] = \sum_{i=1}^{n} \Pr[i \in T \cap \sigma(i) - i > cn],$$

so that

$$\Pr\left[\max_{i \in T} |\sigma(i) - i| > cn\right] \leq 2 \sum_{i=1}^{n} \Pr[i \in T \cap \sigma(i) - i > cn]$$

$$\leq 2 \sum_{i=1}^{n} 2^{-\sqrt{2-n}}(i - F(i/n)) = 2^{-2(1-\sqrt{2-n})} \sqrt{1+o(1)},$$

since $1 - F(c)$ is minimized at $c = 1 - \epsilon$. This concludes the proof of the upper bound.

**Remark:** Strictly speaking, we did not address the bounding of terms for which $i = o(n)$ or $i = n(1 - o(n))$. These may be shown to be negligible compared to our stated upper bound by a slight variation of our argument.

5. PROOF OF THEOREM 2

We analyze an algorithm, which takes advantage of the following decomposition, for any increasing subsequence $t$ of a given permutation $\sigma$:

$$\frac{1}{\tau(\sigma)} = \Pr[T = t] = \prod_{i \in t} \Pr[i \in T \cap |\sigma(i) - i| > cn] \cdot \prod_{i \in t} (1 - \Pr[i \in T \cap |\sigma(i) - i| > cn]),$$

where $\mathcal{F}_{i-1}(t)$ is the event that $T = t \cap [i - 1]$, and we denote by $\mathcal{T}$ a uniformly random increasing subsequence of $\sigma$.

Recall that $\alpha(\sigma, i, j)$ is the number of increasing subsequences of $\sigma$ beginning after $i$, whose initial element $x$ satisfies $\pi(x) > j$, plus 1 for the empty sequence (for the special case of $i = j = 0$, we define $\alpha(\pi, 0, 0) = \tau(\pi)$). Then each conditional probability can be computed as

$$\Pr[i \in T \cap |\mathcal{F}_{i-1}(t)|] = \frac{\alpha(\sigma, i, j)}{\alpha(\sigma, \lambda(i), \lambda(i))},$$

where $\lambda_i(i)$ denotes the largest $j < i$ for which $j \in t$ (when $i = 1$, $\lambda(i) = 0$, and we define $\sigma(0) = 0$ for convenience). Thus, the core of the algorithm is to compute these counts (i.e., $\alpha(\sigma, i, j)$ for $i, j \in [n]$) for a given $\sigma$ by dynamic programming. Essentially, we use the fact that $\alpha(\sigma, i, j)$ can be written recursively as follows:

$$\alpha(\sigma, i, j) = \begin{cases} \alpha(\sigma, i + 1, j) & \text{if } \sigma(i + 1) < j \\ \alpha(\sigma, i + 1, \sigma(i + 1)) + \alpha(\sigma, i + 1, j) & \text{if } \sigma(i + 1) > j \\ \end{cases}$$

**function** $\text{ComputeAlphaTable}(\sigma \in S_n)$

$A \leftarrow \emptyset$

for $i = 1$ to $n$

$A[i, n] \leftarrow 1$

for $i = n - 1$ to $0$

for $j = n - 1$ to $0$

if $\sigma(i + 1) < j$

$A[i, j] \leftarrow A[i + 1, j]$

else

$A[i, j] \leftarrow A[i + 1, \sigma(i + 1)] + A[i + 1, j]$

return $A$

Using this, we can sample $T$ using the following algorithm:

**function** $\text{SampleIncreasingSubsequence}(\sigma \in S_n)$

$A \leftarrow \text{ComputeAlphaTable}(\sigma)$

$t \leftarrow \emptyset$

$\lambda \leftarrow 0$

for $i = 1$ to $n$

With probability $A[i, \sigma(i)]/A[\lambda, \sigma(\lambda)]$, add $i$ to $t$

and set $\lambda = i$.

return $t$

That the algorithm correctly samples from the set of increasing subsequences of $\sigma$ can be verified easily by induction.

Now, computing $A$ takes time $\Theta(n^2)$, and this is followed by a linear number of operations (since $t$ can be represented as a linked list, we need only to append to the end at each step). Thus, the running time is $\Theta(n^2)$, as desired. It is simple to extend this algorithm to higher dimensional analogues (e.g., as in [1]). For dimension $d$, the running time becomes $\Theta((dn)^{d+1})$.

6. REFERENCES


A Large-Scale Network with Moving Servers

Arnab Ganguly  
Louisiana State University  
aganguly@lsu.edu

Kavita Ramanan  
Brown University  
Kavita.Ramanan@Brown.edu

Philippe Robert  
INRIA Paris  
Philippe.Robert@inria.fr

Wen Sun  
INRIA Paris  
Wen.Sun@inria.fr

1. INTRODUCTION

In this note we analyze a queueing network of $N$ queues and $S_N$ servers. The arrival process to each queue is assumed to be Poisson with parameter $\lambda$, and the service times of jobs are independent and exponentially distributed with rate $\mu$. The $S_N$ servers circulate among the $N$ queues. When a server has completed the service of a job at a queue, it chooses at random a non-empty queue without a server if one exists. If all non-empty queues have a server, then the server remains idle. Idle servers are used as soon as an empty queue receive a new job.

This model is a special example of a polling system, for which there is already a huge literature. The main difference with classical models of polling systems is that several servers move around in the network and do not follow a fixed routing schedule among nodes. Furthermore, queues are not located on a circular network. The polling system with moving servers that we study has been used as an idealized model to analyze the performance of Ethernet Passive Optical Networks (EPONs). In this case, queues represent end-users and servers represent the different wavelengths available; see Antunes et al. [1], Boon et al. [2] and Robert and Roberts [9] for example.

As a polling system this special class has received little attention. There are some partial results concerning stability of these systems, by Fricker and Jaibi [5] and Down [4], for example. In this work, we consider the case of a large network, that is, when $N$ goes to infinity while $S_N/N$ converges to some constant.

Stability Condition and Large Scale Behavior

It is assumed that the number of servers $S_N$ is of the order of $\alpha N$, i.e.,

$$\lim_{N \to +\infty} S_N/N = \alpha \in (0, 1).$$

(1)

Clearly a necessary stability condition is that $\lambda N < \mu S_N$. Therefore, it will be assumed that the condition $\rho < \lambda / \mu < \alpha$ holds. We will prove that $\rho < \alpha$ is also a necessary stability condition for all $N$ sufficiently large.

If one considers the total amount of work in the system, as long as there are more than $S_N$ non-empty queues, this system could be seen as an $M/M/S_N$ queue for which the stability condition is well known to be $\rho < \alpha$. The problem in our model is to control the state of the network when the system is congested (that is, has large queues) but works below its maximal capacity, that is, there are less than $S_N$ non-empty queues and thus idle servers. We will use a coupling argument and a quadratic Lyapunov function to resolve this problem.

The stability condition can (roughly) be rephrased as saying that the effective service capacity of each file is $\alpha < 1$. We will show that, under a suitable condition on the initial state, the network behaves on finite time intervals as a set of independent $M/M/1$ queues, each of whose service capacity is 1. A consequence is that, asymptotically, there is a large number of idle servers, of the order of $N$ on any finite time interval. As a result, the sharing of servers among queues increases the effective capacity of each queue of the network. A particular case of this result has been proved in Antunes et al. [1] when the initial states of queues are i.i.d., having a geometric distribution with parameter $\rho$.

2. STOCHASTIC MODEL

Notations. In what follows, for $y > 0$, $(E^n_y)_{n \in \mathbb{N}}$ denotes a sequence of i.i.d. exponential random variables with parameter $y$, and $(N^n_i)_{n \in \mathbb{N}}$ denotes an i.i.d. sequence of Poisson processes with rate $y$.

For $1 \leq i \leq N$ and $t \geq 0$, the state of the system at time $t$ can be represented by $X^N(t) = \{(L_i^N(t), U_i^N(t)), 1 \leq i \leq N\},$ where
- $L_i^N(t)$ is the number of jobs in queue $i$ at time $t$;
- $U_i^N(t) = 1$ if a server is at queue $i$ and $U_i^N(t) = 0$ if not.

Clearly $X^N$ is an irreducible Markov process in the state space

$$\mathcal{S}_N = \left\{ \{(\ell_i, u_i)\} \in \mathbb{N} \times \{0, 1\}^N : \sum_{i=1}^N u_i = \sum_{i=1}^N u_i \mathbb{1}_{\{\ell_i > 0\}} \leq S_N \right\}.$$

For $(a, b) \in \mathbb{N} \times \{0, 1\}$ and $1 \leq i \leq N$, let $e_i(a, b)$ denote the element of $(\mathbb{N} \times \{0, 1\})^N$ whose $i$th coordinate is $(a, b)$ and all other coordinates are $(0, 0)$. One defines, for $x = \{(\ell_i, u_i)\} \in \mathcal{S}_N$,

$$W_N(x) = \sum_{i=1}^N \ell_i \text{ and } A^N(x) = \sum_{i=1}^N u_i \mathbb{1}_{\{\ell_i > 0\}}.$$

(2)

Note that $A^N(x)$ is the number of non-empty queues and $W_N(x)$ is the total number of jobs, in state $x$. For this
Markov process, the transitions from a state $x = (\{(\ell_i, u_i)\}) \in \mathcal{S}_N$ are as follows: for $i = 1, \ldots, N$,

- Arrivals.
  
  $x \mapsto \begin{cases} 
  x + e_i(1, 1), & \text{at rate } \lambda, \\
  x + e_i(1, 0), & \text{at rate } \lambda
  \end{cases}$ if $A^N(x) < S_N$ and $\ell_i = 0$, otherwise.

- Departures.
  
  - If $u_i = 1$, $\ell_i > 1$, then $x \rightarrow x + e_i(-1, 0)$ occurs at rate $\mu$ if $A_N(x) \leq S_N$ and rate $\mu/\mu(A_N(x) - S_N + 1)$ if $A_N(x) > S_N$;
  
  - If $u_i = 1$, $\ell_i = 1$ and $A^N(x) \leq S_N$, $x \rightarrow x + e_i(-1, -1)$ occurs at rate $\mu$;
  
  - If $u_i = 1$ and $A^N(x) > S_N$, then for every $j$ with $u_j = 0$, $\ell_j > 0$, $x \rightarrow x + e_i(-1, 1)$ with rate $\mu/\mu(A_N(x) - S_N + 1)$. 

Let $D_i(0, t)$ denote the number of departures from queue $i$ in the time interval $[0, t]$, and let $(s_{i})_{n \in \mathbb{N}}$ denote the departure instants. Also, let $\xi_i = \mu S_N/(N - S_N + 1)$.

**Proposition 1.** For each $i \in \{1, \ldots, N\}$, there exists a version $X^N$ of the Markov process with initial state $x = (\{(\ell_i, u_i)\}) \in \mathcal{S}_N$ such that $s_i^n \leq t^n$ for all $n \in \mathbb{N}$, where

$$t^n =: \varepsilon_{\mathcal{N}_i}^{n} I_{\{u_i = 0\}} + \varepsilon_{\mathcal{P}}^{n} + \frac{1}{N} \sum_{k=0}^{n} \left( B_{n-k} B_{e_i \mathcal{N}_i} + B_{n-k} B_{e_i \mathcal{P}} \right),$$

with $(B_{n-k})$ denoting i.i.d Bernoulli random variables with

$$\mathbb{P}(B_{n-k} = 0) = \frac{1}{N - S_N + 1}.$$ 

**Proof.** We proceed by induction. Assume $\ell_i > 1$ and let $n = 1$. If $u_i = 1$, then the departure from queue $i$ occurs after an exponential time $\varepsilon^n_{\mathcal{N}_i}$; if $u_i = 0$, then all $S_N$ servers must be busy at other nodes, and one has additionally to wait for a server to move to queue $i$. For this, one of the busy servers has to complete service, which happens at rate $\mu S_N$, and then choose to move to node $i$, which will happen with probability $1/(N - S_N + 1)$. This proves (3) when $n = 1$.

Now, assume $s_i^n \leq t^n$, $n \leq i - 1$. If $A^N(X^n (s_i^n)) \leq S_N$ then, since queue $i$ is not empty and has a server, $s_i^{n+1} = s_i^n + \varepsilon^n_{\mathcal{N}_i}$. Otherwise, one has $A^N(X^n (s_i^n)) > S_N$, either the server leaving node $1$ returns immediately to this node, i.e., $B_{n-k} = 0$, or one has to wait for a time $\varepsilon^n_{\mathcal{P}}$ for the next server to come to node $1$. One concludes that $s_i^{n+1} \leq t_i^{n+1}$. \hfill \Box

**Proposition 2.** Under the condition $\rho < \alpha$, for $N$ sufficiently large, the Markov process $X^N$ is ergodic.

**Proof.** For $t > 0$ and $1 \leq i \leq N$, denote by $\tilde{D}_i(0, t)$ the counting process associated to the sequence $(\varepsilon^n_{\mathcal{N}_i})_{n \in \mathbb{N}}$ of Proposition 1. The renewal theorem, (see, e.g., Grimmett and Stirzaker [6]) and condition (1) imply

$$\lim_{t \to +\infty} \frac{1}{t} \mathbb{E} [\tilde{D}_i(0, t)] = 1 \left( \frac{1}{\mu} + \frac{1}{N - S_N + 1} \right) \sim \mu \alpha,$$

for large $N$. The condition $\rho < \alpha$ gives the existence of $\kappa > 0$, $N_0$ and $T_N < \infty$ such that, for $N \geq N_0$, the relation

$$\frac{1}{T_N} \mathbb{E} \left( N_1^N (T_N) - \tilde{D}_i(0, T_N) \right) \leq - \kappa$$

holds for all $i \in \{1, \ldots, N\}$, where the rate $\lambda$ Poisson process $N_1^N$ represents the arrival process at queue $i$.

Then, given any initial condition $X^N(0) = (\{L^N(0), U^N(0)\}) \in \mathcal{S}_N$, Proposition 1 implies

$$L^N_i(t) \leq \left( L^N_i(0) - \tilde{D}_i(0, t) \right) + N^N_i(t) \left( L^N_i(0) - \tilde{D}_i(0, t) \right) + N^N_i(t),$$

Therefore, $L^N_i(t)^2 - L^N_i(0)^2$ is less than or equal to

$$2(N^N_i(t) - \tilde{D}_i(0, t)) L^N_i(t) + \left( L^N_i(0)-\tilde{D}_i(0, t) \right)^2 + N^N_i(t) (L^N_i(t) - \tilde{D}_i(0, t))^2 \leq 2(N^N_i(t) - \tilde{D}_i(0, t)) L^N_i(t) + 3N^N_i(t)^2 + 3 \tilde{D}_i(0, t)^2.$$

Define, for $x = (\{\ell_i, u_i\}) \in \mathcal{S}_N$, $G(x) = : \ell_1^2 + \ell_2^2 + \cdots + \ell_N^2$. Then, for $t \geq 0$, we have

$$\mathbb{E} [G(X^N(t)) - G(X^N(0))] \leq (\lambda t - \mathbb{E} [\tilde{D}_i(0, t)]) \sum_{i=1}^{N} L^N_i(0) + 3 N \mathbb{E} [N^N_i(t)^2] + \sum_{i=1}^{N} \mathbb{E} [\tilde{D}_i(0, T_N)^2],$$

(4)

From relation (4), one gets that if $G(X^N(0)) \geq K_N$, where

$$K_N = \frac{1}{2 \alpha T_N} \left[ 1 + 3 N \mathbb{E} [N^N_i(0, T_N)^2] + 3 N \mathbb{E} [\tilde{D}_i(0, T_N)^2] \right],$$

then

$$\mathbb{E} [G(X^N(T_N)) - G(X^N(0))] \leq -1.$$

Thus, the function $G$ is a Lyapounov function for the Markov process $X^N$. One concludes that, for $N \geq N_0$, the Markov process $X^N$ is ergodic (see, e.g., [8, Theorem 8.13]). \hfill \Box

3. LARGE SCALE ASYMPTOTICS

We start with a technical result on the duration of time when there is an idle server.

**Lemma 1.** If $\rho < \alpha$ and the initial state $X^N(0) = x_N \in \mathcal{S}_N$ satisfies $A^N(x_N) \geq S_N$ and

$$\limsup_{N \to +\infty} W_N (x_N)/N = C_1 < +\infty,$$

where $A^N(\cdot)$ and $W_N(\cdot)$ are defined by (2), then, if $\tau_N \equiv \inf \{t > 0 : A^N(t) \leq S_N \}$,

there exists $t_0 > 0$ such that

$$\lim_{N \to +\infty} \mathbb{P} (\tau_N \leq t_0) = 1.$$

(5)

**Proof.** On the time interval $[0, \tau_N]$, one has

$$(W_N(X^N(\cdot)) \overset{\text{def}}{=} W_N(x_N) + N^N_1(\cdot) - N^N_1(\cdot))$$

holds. In particular,

$$(W_N(x_N) + N^N_1(\cdot) - N^N_1(\cdot)), \tau_N \geq 0.$$

The law of large numbers for Poisson processes then shows that relation (5) holds with $t_0 = 2 C_1 / (\mu \alpha - \lambda)$. \hfill \Box
The next proposition gives a mean-field result. Specifically, under a stronger condition on ρ, we show that the system behaves as a system of independent M/M/1 queues in which all non-empty queues have a server. Let the processes \{Q_i\}_{i \in \mathbb{Z}} represent the dynamics of i.i.d. M/M/1 queues with arrival rate λ and service rate µ. Then the stability condition \(\rho < \alpha < 1\) implies that each \(Q_i\) is ergodic.

Proposition 3. If the empirical distribution of the N-valued sequence \(\{L_i^N(0)\}_{i=1, \ldots, N}\) converges in distribution to \(\nu\) as \(N \to +\infty\) and if

\[
\rho < \frac{\alpha - \nu(N^*)}{1 - \nu(N^*)},
\]

with \(N^* = N \setminus \{0\}\), then as \(N \to +\infty\),

\[
\left( \frac{1}{N} \sum_{i=1}^{N} \delta_{L_i^N(0), (0)} \right) \to \left( \sum_{m=0}^{\infty} P_{\nu}(Q_1(\cdot) = m) \delta_{(m, 1)} \right),
\]

convergence in distribution of measure-valued process conditioned on the initial distribution \(\nu\) and \(\Rightarrow\) denotes convergence in distribution of measure-valued processes, (see, e.g., [3, 7]).

Proof. For \(1 \leq i \leq N\), set \(Q_i(0) = L_i^N(0)\). Due to the independence of the dynamics of the M/M/1 queues, one has

\[
\left( \frac{1}{N} \sum_{i=1}^{N} \delta_{Q_i(\cdot)} \right) = \left( \sum_{m=0}^{\infty} P_{\nu}(Q_1(\cdot) = m) \delta_m \right),
\]

as \(N \to +\infty\). In particular, this implies that as \(N \to +\infty\),

\[
\left( \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{Q_i(\cdot) > 0\}} \right) \Rightarrow P_{\nu}(Q_1(\cdot) > 0),
\]

where the convergence is of \([0, 1]\)-valued stochastic processes. Note that

\[
\mathbb{P}_{\nu}(Q_1(t) > 0) \leq \nu(0) \mathbb{P}_{\nu}(Q_1(t) > 0) + \nu(N^*) < \alpha,
\]

where the last inequality uses the well known property of the M/M/1 queue that \(t \to Q_1(t)\) is stochastically increasing when \(Q_1(0) = 0\) and the fact that \(\rho\) is the probability that the queue is not empty at equilibrium, and relation (6).

Consequently, for \(T > 0\), if

\[
\mathcal{H}^T_N = \left\{ \sup_{0 \leq t \leq T} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{Q_i(t) \geq S_N\}} \leq \frac{S_N}{N} \right\},
\]

relation (8) shows that \(\lim_{N \to \infty} \mathbb{P}(\mathcal{H}^T_N) = 1\). To conclude the proof, observe that on the event \(\mathcal{H}^T_N\), there are always idle servers up to \(T\), and so the relation

\[
X^N = \left\{ S_i^N, U_i^N \right\}_{i=1, \ldots, N} \overset{\text{dist}}{=} \left\{ Q_i^N(t), \mathbb{1}_{\{Q_i^N(t) > 0\}} \right\}_{i=1, \ldots, N},
\]

holds on the time interval \([0, T]\).

We now obtain a convergence result under a condition on \(\rho\) that does not involve the initial state.

Proposition 4. For any sequence of initial states \(\{x_N\}\) that satisfy

\[
\limsup_{N \to +\infty} \frac{W_N^N(x_N)}{N} < +\infty,
\]

where \(W_N^N(\cdot)\) is defined by relation (2), and if

\[
\rho < 1 - \sqrt{1 - \alpha},
\]

then there exists some \(t_0 > 0\) such that the convergence in (7) holds on any finite time interval after time \(t_0\).

Proof. We use the results and the notations of Lemma 1. There exists some \(t_0\) such that the event \(\mathcal{E}_N = \{\tau_N \leq t_0\}\) has probability close to 1 as \(N\) gets large. Let \(x_N = (L_i^N, u_i^N)\), then by Condition (9), the sequence of empirical distributions of \((\ell_i, i=1, \ldots, N)\) is tight. Since, on the event \(\mathcal{E}_N\),

\[
L_i^N(\tau_N) \leq \ell_i + N_{\tau_N}^N(0, t_0),
\]

the sequence of empirical distributions of \((L_i^N(\tau_N), i=1, \ldots, N)\) is tight and, by definition of \(\tau_N\), any limit point \(\nu\) satisfies the relation \(\nu(N^*) \leq \rho\). The strong Markov property of \(X^N\) applied at time \(\tau_N\) and Proposition 3 give that the convergence in (7) holds on any finite time interval after time \(t_0\) if the relation

\[
\rho < \frac{\alpha - \rho}{1 - \rho} = \frac{\alpha - \nu(N^*)}{1 - \nu(N^*)}
\]

holds, but this is precisely our condition on \(\rho\).

Note that Condition (10) is stronger that the condition \(\rho < \alpha\).

4. Extensions

The main extension currently under study is the model where the servers do not return immediately to the queues but wait for a exponential time with parameter \(\gamma\). The “natural” stability condition in this case is \(\lambda(1/\mu + 1/\gamma) < \alpha\). The analogue of the mean-field result seen in this note would be to state the condition, that under similar initial conditions, after some finite time, a queue in this system behaves as an M/M/1 queue with arrival rate \(\lambda\) and service rate \(\mu\). More interestingly, one would like to show that at the level of the whole network, almost surely there is a finite number of non-empty queues without a server.

5. References


Join-Idle-Queue system with general service times:
Large-scale limit of stationary distributions

Sergey Foss
Heriot-Watt University, Edinburgh, UK
and Novosibirsk State University
s.foss@hw.ac.uk

Alexander L. Stolyar
University of Illinois at Urbana-Champaign
Urbana, IL
stolyar@illinois.edu

ABSTRACT
A parallel server system with n identical servers is considered. The service time distribution has a finite mean 1/µ, but otherwise is arbitrary. Arriving customers are to be routed to one of the servers immediately upon arrival. Join-Idle-Queue routing algorithm is studied, under which an arriving customer is sent to an idle server, if such is available, and to a randomly uniformly chosen server, otherwise. We consider the asymptotic regime where n → ∞ and the customer input flow rate is λn. Under the condition λ/µ < 1/2, we prove that, as n → ∞, the sequence of (appropriately scaled) stationary distributions concentrates at the natural equilibrium point, with the fraction of occupied servers being constant equal λ/µ. In particular, this implies that the steady-state probability of an arriving customer having to wait for service vanishes.

1. INTRODUCTION
We consider a parallel server system consisting of n servers, processing a single input flow of customers. The service time of any customer by any server has the same distribution with finite mean 1/µ. Each customer has to be assigned (routed) to one of the servers immediately upon arrival. This model is sometimes referred to as “supermarket” model.) We study a Join-Idle-Queue routing algorithm, under which an arriving customer is sent to an idle server, if such is available; if there are no idle servers, a customer is sent to one of the servers chosen uniformly at random.

We consider an asymptotic regime such that n → ∞ and the input rate is λn, where the system load λ/µ < 1. Thus, the system remains subcritically loaded. Under the additional assumption that the service time distribution has decreasing hazard rate (DHR), it is shown in [10] that the following property holds.

Asymptotic optimality: As n → ∞, the sequence of the system stationary distributions is such that the fraction of occupied servers converges to constant λ/µ; consequently, the steady-state probability of an arriving customer being routed to a non-idle server vanishes.

The results of [10] apply to far more general systems, where servers may be non-identical. However, the analysis in [10] does rely in essential way on the DHR assumption on the service times; under this assumption the system process has monotonicity property, which is a powerful tool for analysis. Informally speaking, monotonicity means that two versions of the process, such that the initial state of the first one is dominated (in the sense of some natural partial order) by that of the second one, can be coupled so that this dominance persists at all times.

When the service time distribution is general, the monotonicity under JIQ no longer holds, which requires a different approach to the analysis. In the present work we prove the following fact.

Main result (Theorem 2 in Section 2): The asymptotic optimality holds for an arbitrary service time distribution, if the system load λ/µ < 1/2.

We believe that condition λ/µ < 1/2 is purely technical (required for the proof in this paper) and that our main result in fact holds for λ/µ < 1, i.e. as long as the system is stable. This will be discussed in more detail in Section 3.

The key feature of the JIQ algorithm (as well as more general pull-based algorithms [1, 6, 10, 11]), is that it does not utilize any information about the current state of the servers besides them being idle or not. This allows for a very efficient practical implementation, requiring very small communication overhead between the servers and the router(s) [6, 10, 11]. In fact, in the asymptotic regime that we consider, JIQ is much superior to the celebrated “power-of-d-choices” (or Join-Shortest-Queue(d), or JSQ(d)) algorithm [12, 7, 2, 3], in terms of both performance and communication overhead (see [10, 11] for a detailed comparison). The JSQ(d) algorithm routes a customer to the shortest queues among the d servers picked uniformly at random.

We note that when the service time distribution is general, there is no monotonicity under JSQ(d) (just like under JIQ in our case), and this also makes the analysis far more difficult. Specifically, the result for JSQ(d), which is a counterpart of our main result for JIQ, is Theorem 2.3 in [2], which shows the asymptotic independence of individual server states. (Our main result also implies asymptotic independence of server states; see formal statement in Corollary 3.) Theorem 2.3 in [2] imposes even stronger assumptions than ours, namely a finite second moment of the service time and load λ/µ < 1/4 (for non-trivial values of d, which are d ≥ 2); our Theorem 2 only requires a finite first moment of the service time and load λ/µ < 1/2.

In a different asymptotic regime, so called Halfin-Whitt regime (when the system capacity exceeds its load by O(√n)) as opposed to O(n)), and Markov assumptions (Poisson input flows and exponentially distributed service times), JIQ has been recently analyzed in [4, 8]. These papers study diffusion limits of the system transient behavior; Markov...
assumptions appear to be essential for the analysis. Finally, we mention a recent paper [9], which proposes and studies a version of JIQ for systems with packing constraints at the servers.

**Basic notation.** We say that a function is RCLL if it is right-continuous with left-limits. Symbol \( \Rightarrow \) signifies convergence of random elements in distribution. Indicator of event or condition \( B \) is denoted by \( 1(B) \).

### 2. MODEL AND MAIN RESULT

We consider a service system, consisting of \( n \) parallel servers. The system is homogeneous in that all servers are identical, with the same customer service time distribution, given by the cdf \( F(\xi), \xi \geq 0 \). This distribution has finite mean, which WLOG can be assumed to be 1:

\[
\int_0^\infty F^c(\xi) = 1, \quad \text{where } F^c(\xi) = 1 - F(\xi).
\]

Otherwise, the cdf \( F(\cdot) \) is arbitrary. The service queueing discipline at each server is arbitrary, as long as it is work-conserving and non-idling.

Customers arrive as a Poisson process. (This assumption can be relaxed to a renewal arrival process; see Section 4.) The arrival rate is \( \lambda n \), where \( \lambda < 1 \), so that the system load is strictly subcritical.

The routing algorithm is Join-Idle-Queue (JIQ), which is defined as follows. (The JIQ algorithm can be viewed, in particular, as a specialization of the PULL algorithm [10, 11] to a homogeneous system with “single router.”)

**Definition 1 (JIQ).** An arriving customer is routed to an idle server, if there is one available. Otherwise, it is routed to server chosen uniformly at random.

We consider the sequence of systems with \( n \to \infty \). From now on, the upper index \( n \) of a variable/quantity will indicate that it pertains to the system with \( n \) servers, or \( n \)-th system. Let \( W^i_n(t) \) denote the workload, i.e. unfinished work, in queue \( i \) at time \( t \) in the \( n \)-th system. Consider the following fluid-scaled quantities:

\[
x^*_n(t) = (1/n) \sum_i 1\{W^i_n(t) > w\}, \quad w \geq 0.
\]

That is, \( x^*_n(t) \) is the fraction of servers \( i \) with \( W^i_n(t) > w \). Then \( x^*_n(t) = (x^*_n(t), w \geq 0) \) is the system state at time \( t \); \( x^*_n(t) \) is the fraction of busy servers (the instantaneous system load).

For any \( n \), the state space of the process \( x^n(t), t \geq 0 \) is a random element whose distribution is the stationary distribution of the process; in other words, this is a random system state in stationary regime.

The system equilibrium point \( x^* \in \mathcal{X} \) is defined as follows. Let \( \Phi(w) \) denote the complementary (or, tail) distribution function of the steady-state residual service time; the latter is the steady-state residual time of a renewal process with renewal time distribution function \( F(\cdot) \). We have

\[
\Phi(w) = \int_0^\infty F^c(\xi)d\xi, \quad w \geq 0.
\]

Then,

\[
x^* = (x^*_w = \lambda \Phi(w), w \geq 0) \in \mathcal{X}.
\]

In particular, the equilibrium point is such that “the fraction of occupied servers” \( x^*_w = \lambda \). Our main result is the following

**Theorem 2.** If \( \lambda < 1/2 \), then \( x^*(\infty) \Rightarrow x^* \) as \( n \to \infty \).

The detailed proof of Theorem 2 can be found in [5]. In this short paper we will only give (in Section 3) a high level description of the proof approach and the intuition for constructing the proof. (The proof is straightforward to verify. In other words, the process has unique stationary distribution.)

### 3. PROOF APPROACH AND DISCUSSION OF CONDITION \( \lambda < 1/2 \)

The approach we use to establish the convergence of stationary distributions in Theorem 2 is as follows. (The detailed proof is in [5].) We find a set \( A \subset \mathcal{X} \) and a fixed finite time \( T \), such that, with high probability, for all large \( n \), (a) \( x^n(\infty) \in A \) and (b) \( x^n(0) \in A \) implies that \( x^n(T) \) is close to \( x^* \). Property (b) is key. When \( n \) is large, the trajectory \( x^n(t) \) is almost deterministic. (In fact, the problem reduces to the analysis of “fluid limit” trajectories, which are the limits of \( x^n(t) \) as \( n \to \infty \).) Then, informally speaking, property (b) above reduces to the property (b’): trajectories \( x^n(t) \) converge to \( x^* \) as \( t \to \infty \). The absence of process monotonicity (described in Section 1) makes proving (b’) difficult. We now describe – very informally – the key idea, which we use in our proof of convergence (b’), and which relies on the condition \( \lambda < 1/2 \).
Suppose $n$ is large. Consider an initial state $x_0^n(0)$, such that the total amount of (fluid-scaled, i.e. multiplied by $1/n$) unfinished work is upper bounded by $C < \infty$. Pick $\alpha$ such that $\alpha > \lambda$ and $\alpha + \lambda < 1$; this can be done if and only if $\lambda < 1/2$. Then, at some finite time $\tau$, the system must reach a state with $\alpha n$ servers being idle. (Otherwise, if at least $(1 - \alpha)n$ servers would continue to be busy as time goes to infinity, the unfinished work would become negative, since $1 - \alpha > \lambda$.) Denote by $S_n$ the set of those $\alpha n$ servers, which are idle at time $\tau$. Starting time $\tau$, WLOG, assume that all new arriving customers go to an idle server in $S_n$, as long as there is one available. Consider the subsystem, consisting only of the servers in $S_n$; starting time $\tau$ and until the (random) time when all servers in $S_n$ become busy, the behavior of this subsystem is obviously equivalent to that of the infinite-server system, $M/GI/\infty$, with idle initial state. If $n$ is large, the behavior of $x^n(t)$ for such $M/GI/\infty$ system is “almost deterministic” and such that the (scaled) number of occupied servers $x^n_0(t)$ in it is “almost monotone increasing, converging to $\lambda < \alpha^\prime$ and, moreover, $x^n(t)$ “converges” to $x^\prime$. But this means that after time $\tau$ the subsystem $S_n$ will “always” have idle servers, which in turn means that its state will “converge” to $x^\prime$ as $t \to \infty$. Also, after time $\tau$, the subsystem consisting of the servers outside $S_n$ will “never” receive any new arrivals and will “eventually” empty. Thus, $x^n(t)$ for our entire system “converges” to $x^\prime$.

Turning the key intuition, described above informally, into a formal proof is the subject of paper [5]. Set $A \in \mathcal{X}$ is picked by using a constructed uniform in $n$ upper bound on the stationary distribution of the workload of an individual server. The states in $A$ are such that the total (scaled) workload is not necessarily upper bounded by a constant $C$ (in fact, if the second moment of the service time is infinite, the steady-state total workload in the system is infinite with probability 1); however, for states in $A$ the (scaled) workload is bounded by $C$ on a close-to-1 fraction of servers – this suffices for the proofs. The property (b’) is proved uniformly for fluid limits starting from $A$ – from here we obtain that (b) holds for the pre-limit processes with high probability, uniformly for all large $n$.

As explained above, the proof of Theorem 2 relies in essential way on condition $\lambda < 1/2$. However, we believe that this condition is purely technical, and Theorem 2 in fact holds for any $\lambda < 1$. Establishing this fact will most likely require a different proof approach, although some elements of the analysis in [5] may turn out to be useful for the proof of a more general result.

4. GENERALIZATIONS

The following generalizations of Theorem 2 hold. Here we only state them – see [5] for a detailed discussion.

Renewal arrival process. Theorem 2 and its proof easily generalize to the case when the arrival process is renewal; namely, when in the $n$-th system the interarrival times are i.i.d., equal in distribution to $A/n$, where $A$ is a positive random variable, $E A = 1/\lambda$. (Mild assumptions on the interarrival time distribution are needed. For example, it suffices that this distribution has an absolutely continuous component.) The common process state space contains an additional scalar variable $u$, which is the residual interarrival time; clearly $u^n(\infty) \Rightarrow 0$ as $n \to \infty$. The more general form of Theorem 2 is as follows:

If $\lambda < 1/2$, then $(u^n, x^n)(\infty) \Rightarrow (0, x^\prime)$.

Biased routing when all servers are busy. The specific rule – uniform at random – for routing arriving customers when all servers are busy, can be relaxed as follows. It suffices that the arrival rate into a server when it is busy is upper bounded by some $\bar{\lambda} < 1$; this holds, for example, when the probability that any busy server receives an arrival does not exceed $(1/n)(\bar{\lambda}/\lambda)$ for some $\lambda < 1$. When this condition holds – in addition to $\lambda/\mu < 1/2$ – Theorem 2 holds as is.

Finite buffers. Suppose, we allow some or all servers to have finite buffers (of same or different sizes). If a server has finite buffer of size $B \geq 1$, and already has $B$ customers, any new customer routed to to this server is blocked and leaves the system. For such more general system Theorem 2 holds as is.

5. REFERENCES


1. INTRODUCTION

Epidemic-like stochastic processes with time-varying parameters have been introduced to model viral behaviors in communication and social networks, cybersecurity systems, financial markets, and so on [5]. We additionally observe that such stochastic processes can include interactions between local (micro) and global (macro) behaviors within the process, involving migration from one operating regime to another and transitions within each regime. To address this phenomena, we consider herein nearly completely decomposable (NCD) structures [3] in epidemic-like stochastic processes with time-varying behaviors. In doing so, we extend known results for NCD processes to include time-varying parameters and we devise mean-field limits of such epidemic-like processes as regime-switching dynamical systems.

Specifically, in a previous study [5], we explored the connection between a general class of epidemic-like stochastic processes with time-varying parameters that characterizes the type of viral behaviors exhibited in environments such as communication and social networks. We established that, under general assumptions on the time-varying process and under a mean-field scaling with respect to population size \( n \), the stochastic process converges to a time-varying dynamical system, extending the corresponding results in [4] to time-varying processes. In this paper, we first extend our previous work to study the stationary behavior of both the original stochastic process and the mean-field limiting dynamical system, and verify that they, in fact, have similar asymptotic behavior with respect to time. In other words, we establish that the following diagram is commutative:

\[
\begin{array}{ccc}
\lim_{n \to \infty} X_n(t) & \xrightarrow{X(t)} & \lim_{n \to \infty} X_n \\
\downarrow & & \downarrow \\
\lim_{n \to \infty} X(t) & \xrightarrow{X} & X
\end{array}
\]  \tag{1}

Given our above observation on local and global behaviors, we further study a general class of NCD Markov chains (MCs) with time-varying parameters and the corresponding mean-field limiting systems. The classical approach of Courtois consists of approximating the stationary distribution of an NCD MC by a completely decomposable (CD) MC, which characterizes the dynamics within each regime in isolation, together with an auxiliary MC that governs the transitions between regimes [3]. Both the CD MC and the auxiliary MC are mathematically more tractable than the original NCD MC, and the approximation error is shown to be bounded by the deviation of the transition matrix of the NCD MC from that of the CD MC, which is of order \( O(\epsilon) \).

In this paper, we also streamline the NCD analysis, extending these results to sequences of NCD MCs associated with epidemic-like stochastic processes. This allows us to further approximate such large-scale NCD stochastic processes with time-varying parameters by the corresponding mean-field limiting systems, thus establishing that diagram (1) above is also commutative with \( X := (X_n(t), X(t), X_n, X) \) replaced by \( X_n := (X_n(t), X(t), X_n, X_n) \).

2. MEAN-FIELD LIMIT AND STATIONARY DISTRIBUTION

Following [5], we consider a sequence of Markov processes \( \hat{Z}_n = \{(\hat{X}_n(t), \hat{Y}_n(t)); t \geq 0\} \) indexed by the population size \( n = 1, 2, \ldots \) and defined over the probability space \((\Omega_n, \mathcal{F}_n, \mathbb{P}_n)\), comprised of state space \( \Omega_n := \{(i, j) : 0 \leq i, j \leq n, i + j = n\} \), \( \sigma \)-algebra \( \mathcal{F}_n \) and probability measure \( \mathbb{P}_n \), with initial probability distribution \( \mathbf{\alpha}_n \). The ordered pair \( (\hat{X}_n(t), \hat{Y}_n(t)) \) denotes the non-infected and infected amount of the population at time \( t \), respectively, where we assume the population of size \( n \) is connected through a complete graph. Let the states \( i \in \{0, 1, 2, \ldots, n\} \) represent the number of non-infected members of the population. Then the transition probabilities are given by \( p_{i, i+1} = \frac{(n-1)\mu}{\nu} \) and \( p_{i, i-1} = \frac{\mu}{\nu} \) for all \( i = 0, 1, \ldots, n \). For state \( i = n \), i.e., everyone is healthy, there is a transition back into the state \( n-1 \) with rate \( \nu \), which guarantees the existence of a stationary distribution.

To start, it is easy to derive the stationary distribution from the balance equations. For \( i = 0, 1, \ldots, n-1 \), we have

\[
\pi_i = \frac{n+1}{i(n-i)} \left( \frac{\mu}{X} \right)^i \pi_0, \tag{1}
\]

together with

\[
\pi_n = \frac{n^n}{(n-1)!} \left( \frac{\mu}{X} \right)^{n-1} \left( \frac{\mu}{\nu} \right) \pi_0. \tag{2}
\]

Remark. We know that the distribution is roughly a truncated Poisson distribution with rate \( n \frac{\mu}{X} \). In order to apply Markov’s inequality argument, we need to estimate the first two moments. Note that we only need to estimate their order in \( n \), so we can simply assume that \( \pi_n \) will have the same format as \( \pi_i, i = 0, 1, \ldots, n-1 \); later we can include the difference, which will not change our main results.
For the first moment, we have
\[
E[X_n] = \sum_{i=1}^{n} i \pi_i = \sum_{i=1}^{n} \frac{n}{i-1}(n-i) \left( \frac{\mu}{\lambda} \right)^i \pi_0
\]
\[
= n \left( \frac{\mu}{\lambda} \right)^n \sum_{i=1}^{n-1} \frac{n-i}{n-i+1} \pi_i,
\]
from which we can conclude \(E[X_n] = n \frac{\mu}{\lambda} + o(n)\). Meanwhile,
\[
E[X_n - E[X_n]]^2 = E[X_n^2] - E[X_n]^2
\]
\[
= \sum_{i=1}^{n} i^2 \pi_i - \left( \sum_{i=1}^{n} i \pi_i \right)^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} i(j-i) \pi_j \pi_i,
\]
and thus we have \(\text{Var}[X_n] = O(n)\). Therefore, for any \(A > 0\),
\[
P \left( \left| \frac{X_n}{n} - \frac{\mu}{\lambda} \right| > A \right) = E \left[ \left( X_n - n \frac{\mu}{\lambda} \right)^2 > n^2 A^2 \right]
\]
\[
\leq \frac{E \left[ X_n - n \frac{\mu}{\lambda} \right]^2}{n^2 A^2}.
\]
From this derivation we know that the numerator is of order \(O(n)\), hence the above quantity will converge to zero as \(n \to \infty\). This implies that \(\bar{X}_n\) converges weakly to \(\frac{\mu}{\lambda}\) as \(n \to \infty\).

To complete our analysis of diagram \(\{1\}\), following \([5]\), we show that the sequence of MCs, when scaled by \(n\), converges to a dynamical system described as follows
\[
\frac{dx}{dt} = -\lambda(t)x + \mu(t)y \quad \text{and} \quad \frac{dy}{dt} = \lambda(t)x - \mu(t)y.
\]
Furthermore, we show that all trajectories of the dynamical system will approach time-varying asymptotic states when \(\lambda(t)\) and \(\mu(t)\) are slowly varying, which includes as a special case \(\mu(t) = c \in (0,1)\) as \(t \to \infty\). It is easy to see that the stationary distribution of the inhomogeneous MC \(X_n(t)\) is actually the same as that of a homogeneous MC with \(\lambda(t)\) and \(\mu(t)\) replaced by constants \(\lambda\) and \(\mu\), such that \(\frac{\mu}{\lambda} = c\). The above arguments therefore demonstrate that, if we first let the time parameter \(t\) go to infinity and then scale by \(n \to \infty\), we will get the same point measure, and thus diagram \(\{1\}\) is commutative.

3. ANALYSIS OF NCD SYSTEMS

A primary result of Courtois \([2]\) is that the stationary distribution of an NCD can be accurately approximated by a system comprising an auxiliary MC and a CD MC. Our first result shows that, under mild conditions, this is true even when we have a sequence of NCD MCs indexed by \(n\), for which an error bound uniform in \(n\) can also be established. Consequently, in the limit as \(n \to \infty\), we obtain an MC whose state space consists of several point measures such that this MC has bounded error in addition to the sequence of NCD MCs. Our ultimate goal will be to establish the commutativeness of diagram \(\{1\}\) with \(X\) replaced by \(X_c\).

Consider a two-parameter process, \(X_{n,t}\), representing the underlying MC that characterizes the discrete system which not only exhibits stochastic epidemic-type behavior within each regime, but also transitions between different regimes. Here, \(n\) is a parameter related to space, often representing the total population of the system under consideration, and \(\epsilon\) represents the relative magnitude of the transition rates between the different regimes.

More specifically, the transition probability matrix for \(X_{n,t}\) will be denoted as \(Q_{n,t}\), where we have
\[
Q_{n,t} = Q_n + \epsilon C_n
\]
and
\[
Q_n = \begin{bmatrix}
Q_1 & Q_2 & \cdots & 0 \\
0 & \ddots & \cdots & \vdots \\
\vdots & \cdots & \ddots & \vdots \\
0 & \cdots & 0 & Q_M
\end{bmatrix}.
\]
The \(n \times n\) matrix \(Q_n\), for \(m = 1, 2, \ldots, M\), represents the transition probabilities for the states within regime \(m\), and \(C_n\) represents the transition probabilities between different regimes, whose row sums will be bounded by one. For ease of exposition, we will suppress the index \(n\), but will address specific details with respect to \(n\) when necessary.

First, define \(\Lambda'(m) := \text{diag}(\lambda_{mi})\) to be the diagonal matrix of the eigenvalues of \(Q_{n,m}\), and \(\Lambda'^* := \text{diag}(\Lambda'(m))\). We then know there exists a non-singular matrix \(H\) such that \(H^{-1}Q'H = \Lambda'^*\), and thus \(H^{-1}QH = \Lambda'^* + eH^{-1}CH\). Next, let \(B\) denote the matrix whose rows are the left eigenvectors of \(H^{-1}QH\). Upon multiplying the above equality by \(B\), we have \(AB = BA\Lambda'^* + bB^{-1}CH\). Observe that, for each \(j = 1, 2, \ldots, n\), \(AB_{(j)} = BA_{(j)} + eB^{-1}CH_{(j)}\), especially for each \(j = n + 1, m = 0, 1, \ldots, M - 1\). Then, from this we can obtain the \(O(\epsilon)\) order for some elements in \(B\).

3.1 Auxiliary MC

Let \(\pi_j^*\) denote the stationary distribution of \(Q^*_1\), i.e., solution of \(Q^*_1 \pi_j^* = \pi_j^* Q^*_1\). The state space of the auxiliary MC consists of the set of indices for the blocks, i.e., the number of states is \(M\). The transition probabilities are defined by \(q_{km} = \sum_{i \in (k)} \pi_i^* \sum_{j \in (m)} p_{ij}\) for \(k, m = 1, 2, \ldots, M\), where, for any \(m, b(m)\) represents the set that contains all the indices in block \(m\). Intuitively, this is the aggregation, by weights of \(\pi^*\) (stationary under \(Q^*_1\)), of the probability mass of transitions between blocks, which is referred to as the macro-variable system through variable aggregation in \([2]\).

3.2 Error Analysis

Suppose \(\bar{\pi}\) is the stationary distribution of the auxiliary MC with \(M\) states. We further have as above that \(\pi_j^*\) is the stationary distribution conditional on being in regime \(i\). Let \(\pi\) denote the stationary distribution of the original NCD MC. Then the quantity \(\bar{\pi}_i \pi_{ij}^*\) will be used as an approximation of \(\pi_{ij}\), namely the stationary probability for state \(j\) in regime \(i\) of the original NCD MC.

For the (transposed) stationary vector, we know that \(\pi = B(1)H^{-1}\), where \(B(1)\) denotes the first row of the matrix \(B\), i.e., the left eigenvector of the matrix \(H^{-1}QH\) corresponding to the eigenvalue \(1\). Recall that \(\pi\) can be viewed as the left eigenvector of \(Q\) corresponding to the eigenvalue of \(1\), and \(B(1)\) represents the left eigenvector of \(H^{-1}QH\) corresponding to the same eigenvalue.

From the fact that \(H^{-1}\) is a block matrix, we can rewrite the above into the following block form
\[
\bar{\pi}_m = B(1)_m H_m, \quad m = 1, 2, \ldots, M,
\]
where \(B(1)_m\) denotes the \(m\)-th block of the (transposed) vector \(B(1)\). For all vectors and matrices with subscript \(m\), we refer to the \(m\)-th block of the vectors and matrices.
Next, we have
\[ \pi_m = B(1)_m H_m^{-1} = B(1)_m I_{m,m} H_m^{-1} + B(1)_m I_{m,h} H_m^{-1}. \]
Note that \( i_m \) is the cardinality of \( b(m) \), and \( I_{m,m} \) denotes the
identity matrix of dimension \( i_m \times i_m \). Let \( I_{m,h} \) denote the
matrix whose first row and first column element is 1 with
all others 0, and \( I_{m,m} \) the difference between \( I_{m,m} \) and \( I_{m,h} \).

We therefore obtain
\[ \pi_m^t - \tilde{\pi}_m \pi_t^* = B(1)_m I_{m,m} H_m^{-1} + [B_m I_{m,h} H_m^{-1} - \tilde{\pi}_m \pi_t^*], \]
for which we have the following main result.

**Theorem 3.1.** The approximation error is of linear order
of the NCD parameter \( \epsilon \), namely
\[ \pi_m^t - \tilde{\pi}_m \pi_t^* = O(\epsilon). \]

We now establish Theorem 3.1 by showing that both terms
on the RHS of Eq. (3) are \( O(\epsilon) \).

### 3.2.1 Estimation of the first term

Once again, the basic identity is given by \( AB = BA^* + \epsilon BH^{-1}CH \).
For convenience, let us denote the set \( F \subseteq \{1, 2, \ldots, n\} \) containing all the indices that are the first
of each block, i.e., \( F := \{ \sum_{i=1}^n i_k + 1, m = 0, 1, \ldots, M - 1 \} \).
Let \( E \) be the matrix containing \( e_j, j \in F \). We then have
\[ AB - BA^* E = \epsilon BH^{-1}CH. \]
and thus all the elements in the matrix \( AB - BA^* E \) are
\( O(\epsilon) \). It follows that in each block the off-diagonal elements
on the first row and column will be \( O(\epsilon) \), and hence these
B elements will be \( O(\epsilon) \). At this point, we require that
there exist a nonzero uniform lower bound for the spectral
gaps of the sequence of MCs, indexed by \( n \). Meanwhile,
Meise [6] shows that such lower bounds can be connected to
a supremum of a family of expected hitting times. It is easy
to verify for the epidemic model, under mild conditions
on \( \lambda(t) \) and \( \mu(t) \), that the supremum of this family of expected
hitting times can be estimated as \( O(n) \), which implies that
a lower bound on the spectral gap is \( O(1/n) \). With this
fact, we can conclude that every nonzero element in \( B(1)_m I_{m,m} \)
will be \( O(\epsilon \sqrt{n}) \), and so is the first term in (3).

### 3.2.2 Estimation of the second term

Since \( I = E + (I - E) \), we have
\[ AB - BA^* E = \epsilon BH^{-1}CH. \]  

From the above analysis, each term in \( B1'_L \) is \( O(\epsilon \sqrt{n}) \), and
hence the RHS of Eq. (6) is \( O(\epsilon \sqrt{n}) \). The LHS is basically
\( P - I \), where \( P \) is the transition matrix for the auxiliary MC.
The desired result can be obtained after we add \( \bar{\pi}(P - I) \) to
the LHS of the equation, which would not change the
equation, since \( \bar{\pi}(P - I) = 0 \). Observe that \( HC \) is basically
a vector whose elements are all ones, which is due to the fact
that on each block this is simply the first right eigenvector,
and thus an all-ones vector due to stochasticity. Therefore,
\( \epsilon BH^{-1}CH \) actually recovers the dynamics of the auxiliary MC.

**Remark.** While we will only study the class of epidemic-like
systems in what follows, the error analysis presented
here applies to general NCD MCs.

### 3.3 Complete the Diagram

Recall that \( X_{n,t} \) represents the stationary distribution of
\( X_{n,t} \), or the limit as \( t \to \infty \). Let \( Y_{n,t} \) denote the stationary
distribution for the CD MC and auxiliary MC. The above arguments confirm that the total variation distance
between the two sequences of measures are of order \( O(\epsilon \sqrt{n}) \),
i.e., \( d_{TV}(X_{n,t}, Y_{n,t}) = O(\epsilon \sqrt{n}) \), uniformly in \( n \). Next, consider
\( X_t = \frac{1}{n} X_{n,t} \) and \( Y_t = \frac{1}{n} Y_{n,t} \). We further conclude
that \( d_{TV}(X_t, Y_t) = O(\epsilon) \), uniformly in \( n \). Meanwhile, from
our analysis in Section 2, we know that \( Y_t \) will be an MC be-
tween point measures, which provides a nice interpretation
of the original stochastic process.

Let us now take the other route in the alternative version
of diagram (1). First, consider the process \( X_t(t) = \frac{1}{n} X_{n,t} \) and \( Y_t(t) = \frac{1}{n} Y_{n,t} \). From the analysis in [5], we
know that \( Y_t(t) \) is a regime-switching dynamical system, a
special case of the broader class of hybrid dynamical sys-
tems which have many applications in modeling biological
and chemical systems [1]. We can also apply the analysis of
the dynamical system in [5] to conclude that the equilibrium
state of \( Y_t(t) \) will be those point measures in combination.
Hence, we conclude that diagram (1) with \( X_t \) replaced by
\( X_t \), is also commutative.

### 4. REFERENCES

systems and its applications to biological and medical
Decomposable Stochastic Systems, *Econométrica*,
control of epidemic-like stochastic processes with
chain via hitting times and coupling, *Journal of
nonautonomous chaotic systems without

---

1Even simple regime-switching dynamical systems can exhibit
complicated dynamics. For instance, a simple 2-
dimensional dynamical system with a continuous piecewise-
linear vector field and driven by a periodic signal, can exhibit
chaotic behavior [7].