# Work Capacity of Freelance Markets: Fundamental Limits and Decentralized Schemes

1

Avhishek Chatterjee, Lav R. Varshney, and Sriram Vishwanath

# Abstract

Crowdsourcing of jobs to online freelance markets is rapidly gaining popularity. Most crowdsourcing platforms are uncontrolled and offer freedom to customers and freelancers to choose each other. This works well for unskilled jobs (e.g., image classification) with no specific quality requirement since freelancers are functionally identical. For skilled jobs (e.g., software development) with specific quality requirements, however, this does not ensure that the maximum number of job requests is satisfied. In this work we determine the capacity of freelance markets, in terms of maximum satisfied job requests, and propose centralized schemes that achieve capacity. To ensure decentralized operation and freedom of choice for customers and freelancers, we propose simple schemes compatible with the operation of current crowdsourcing platforms that approximately achieve capacity. Further, for settings where the number of job requests exceeds capacity, we propose a scheme that is agnostic of that information, but is optimal and fair in declining jobs without wait.

#### **Index Terms**

freelance markets, capacity, queuing theory, decentralized algorithms

#### I. Introduction

Methods and structures for information processing have been changing. Enabled by the proliferation of modern communication technologies, globalization and specialization of workforces has led to the emergence of new decentralized models of informational work. Moreover, the millennial generation now entering the workforce often favors project-based or job-based work, as in crowdsourcing and social production [2], [3], rather than long-term commitments [4]. Indeed over the last decade, more than 100 'human clouds' have launched with a variety of structures. These platforms serve clients by harnessing external crowds, and global enterprises similarly harness their internal crowds [5]–[7], making use of human cognitive surplus for information processing [8].

Platforms follow different collective intelligence models [9], [10], and require different strategies for allocating informational work to workers. In crowdsourcing contest platforms like InnoCentive and TopCoder, there is *self-selection*: work is issued as an open call and anyone can participate in any job; the best submission wins the reward [11]–[14]. In microtask crowdsourcing platforms like Amazon Mechanical Turk, any worker is assumed able to do any job and so first-come-first-serve strategies are often used; level of reliability may be considered in optimal allocation [15]. In freelance markets like oDesk and Elance, however, specialized jobs must be performed by skilled workers: allocation requires careful selection from the large pool of variedly-skilled freelancers.

Freelance market platforms serve as spot markets for labor by matching skills to tasks, often performing ondemand matching at unprecedented scales. For example, oDesk had 2.5 million workers and nearly 0.5 million clients in 2013 [10]. Herein we study allocation and scheduling of informational work within these kinds of platforms, via a queuing framework. We aim to establish fundamental limits through a notion of *work capacity*, and also develop decentralized algorithms, which are easily-computed, that nearly achieve these performance limits.

Freelancers may have one or more skills (that are known, cf. [16]) and jobs may have multiple parts, called *tasks*, that require separate skills. Due to job skill requirement variety and limited freelancer ability, it is often not possible

Part of the material in this paper was presented at IEEE INFOCOM, Hong Kong, April-May 2015 [1].

- A. Chatterjee is with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712 USA. (e-mail: avhishek@utexas.edu).
- L. R. Varshney is with the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA. (e-mail: varshney@illinois.edu).
- S. Vishwanath is with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712 USA. (e-mail: sriram@austin.utexas.edu).

to find a freelancer that meets all requirements for a job: a job may have to be divided among freelancers. Moreover, a task in a job may require so much time that even the task may have to be divided among multiple freelancers. There are reputation systems within freelance market platforms, so freelancers have a reputation level as well as minimum acceptable hourly rate and skills, which allow worker categorization. Some freelancers are adaptable in terms of hours available to spend on a particular type of task, whereas others pre-specify hours available for each kind of task. Here we consider the non-adaptable setting where, for example, a freelancer may be available for 20 hours (per week) of any C++ or Java programming, or may be available for 10 hours (per week) of C++ and 10 hours of Java. Studying limits for adaptable freelancers and designing centralized schemes (and their approximations) are similar, but the distributed schemes require a different approach.

The objective of the platform is to find a good allocation of jobs (and tasks) to freelancers. Since working on a task requires synchronization among freelancers, work can only start when the whole task has been allocated. On the other hand, for some jobs there are interdependencies between different tasks [17] and hence, for these jobs all tasks must be allocated before the job starts. Moreover, some jobs may require all parts to be done by freelancers with the same level of expertise for uniform quality and money spent. These considerations lead to concepts of decomposability and flexibility that are central to our development.

In the ethos of self-selection, it is desirable for crowd systems to not be centrally controlled, but rather for jobs and freelancers to choose each other. Currently, this may happen randomly or greedily. This is clearly not optimal, as the following example illustrates. Consider two types of jobs (single task) and two categories of freelancers. A type 1 job can be served by either of the worker categories (example, lower reputation requirement) whereas type 2 jobs can only be served by category 2 workers. If freelancers and jobs are allocated arbitrarily then it may happen that type 1 uses many category 2 freelancers and many type 2 jobs remain unserved.

Optimal centralized allocation of informational tasks under the constraints of crowd systems is related to hard combinatorial problems such as the knapsack problem. Compared to scheduling problems in computer science [18], communication networks [19], [20], and operations research [21], crowd systems face challenges of freedom of self-selection, need for decentralized operation, and uncertainty in resource availability.

Prior works in the information theory, networking, and queueing literatures are similar to our work in terms of theoretical framework, performance metrics, and the nature of performance guarantees, but are not directly related. The notion of *capacity* of a resource-shared system where jobs are queued until they are served and the notion of a capacity-achieving resource allocation scheme for this kind of system came to prominence with the work of Tassiulas and Ephremides [22], [23]. The capacity concept and capacity-achieving schemes were subsequently developed for applications in communication networks [19], [20], [24], [25], cloud computing [26], online advertising [27], [28], and power grids [29], among others. With the advent of cloud services, large-scale systems have attracted significant research interest: resource allocation schemes and their performance (queueing delays, backlogs, etc.) in the large-scale regimes have been studied [30]–[33].

In this paper, our goal is to understand the fundamental limits (capacity) of freelance markets and ways to achieve this ultimate capacity. We first develop a centralized scheme for achieving these maximum allocations where a central controller makes all job allocation decisions. Given the potential large scale of platforms, we also discuss low-complexity approximations of the centralized scheme that almost achieve the limit. Finally, with an eye towards giving flexibility to customers (job requesters) in choosing freelancers, we propose simple decentralized schemes with minimal central computation that have provable performance guarantees. Further, since job arrival and freelancer availability processes are random (and sometimes non-stationary), we also address ways to adapt when the system is operating outside its capacity limits.

# II. SYSTEM MODEL

We first provide formal definitions of the nature of informational work and workers, and establish notation.

Freelancers (or agents) are of L categories. In each category  $l \in [L]$ , there are  $M^l$  types of agents depending on their skill sets and available hours. There are S skills among agents of all categories and types. An agent of category l and type i has a skill-hour vector  $h_i^l$ , i.e.  $h_{i,s}^l$  available hours for work involving skill  $s \in [S]$ .

Jobs posted on the platform are of N types. Each type of job  $j \in [N]$  needs a skill-hour service  $r_j$ , i.e.  $r_{j,s}$  hours of skill s. A part of a job of type j involving skill s is called a (j,s)-task if  $r_{j,s} > 0$ , which is the size of this task.

A job of type j can only be served by agents of categories  $l \in \mathcal{N}(j) \subset [L]$ . This restriction is captured by a bipartite graph G = ([N], [L], E), where a tuple  $(j, l) \notin E \subset [N] \times [L]$  implies that category l agents cannot serve jobs of type j.

On the platform, jobs are allocated at regular time intervals to available agents, these epochs are denoted by  $t \in \{1, 2, \ldots\}$ . Jobs that arrive after epoch t has started are considered for allocation in epoch t+1, based on agents available at that epoch. Unallocated jobs (due to insufficient number of skilled agents) are considered again in the next epoch.

Jobs arrive according to a  $\mathbb{Z}_+^N$ -valued stochastic process  $\mathbf{A}(t) = (A_1(t), A_2(t), \dots, A_N(t))$ , where  $A_j(t)$  is the number of jobs of type j that arrive in scheduling epoch t.

The stochastic process of available agents at epoch t is  $\mathbf{U}(t) = (\mathbf{U}^1(t), \mathbf{U}^2(t), \dots, \mathbf{U}^L(t))$ . For each agent category l,  $\mathbf{U}^l(t) = \left(U^l_1(t), U^l_2(t), \dots, U^l_{M^l}(t)\right)$  denotes the number of available agents of different types at epoch t.

We assume processes  $\mathbf{A}(t)$  and  $\mathbf{U}(t)$  are independent of each other and that each of these processes is independent and identically distributed for each t.<sup>1</sup> We also assume that each of these processes has a bounded (Frobenius norm) covariance matrix. Let  $\Gamma(\cdot)$  be the distribution of  $\mathbf{U}(t)$ , and let  $\lambda = \mathbb{E}[\mathbf{A}(t)]$  and  $\mu^l = \mathbb{E}[\mathbf{U}^l(t)]$  for  $l \in [L]$  be the means of the processes.

At any epoch t, only an integral allocation of a task (say (j,s)) is possible. A set of tasks  $t_1,t_2,\ldots,t_n$  of size  $r_1,r_2,\ldots,r_n$  of skill s can be allocated to agents  $1,2,\ldots,m$  only if available skill-hours for skill s of these agents  $h_1,h_2,\ldots,h_m$  satisfy

$$\sum_{n=1}^{n} v_{ip} \le h_i, \sum_{q=1}^{m} v_{qj} \ge r_j, j \in [n], i \in [m]$$

for some  $\{v_{pq} \geq 0\}$ .

Whether different tasks of a job can be allocated at different epochs and across different categories of agents depend on the type of the job.

**Definition 1.** A type of job j is called non-decomposable (decomposable) if different tasks comprising it are (are not) constrained to be allocated at the same epoch.

**Definition 2.** A type of job j is called inflexible (flexible) if different tasks as well parts of tasks comprising it are (are not) constrained to be allocated to the same category of agents.

In a system with only decomposable jobs, given a set of  $\{\mathbf{u}^l = (u^l_1, u^l_2, \dots, u^l_{M^l}) : l \in [L]\}$  agents (that is,  $u^l_i$  agents of category l and of type i within that category), a number  $a_{j,s}$  of (j,s)-tasks can be allocated only if there exist non-negative  $\{z^l_{j,s} : (l,j,s) \in [L] \times [N] \times [S]\}$  satisfying

$$\sum_{l} z_{j,s}^{l} = a_{j,s}, z_{j,s}^{l} = 0 \text{ if } (j,l) \notin E, \text{ for all } j \in [N], s \in [S],$$

$$\sum_{j \in [N]} z_{j,s}^{l} r_{j,s} \leq \sum_{i \in [M^{l}]} u_{i}^{l} h_{i,s}^{l}, \text{ for all } l \in [L], s \in [S].$$
(1)

On the other hand, given a set of  $\{\mathbf{u}^l = (u^l_1, u^l_2, \dots, u^l_{M^l}) : l \in [L]\}$  agents in a system with only non-decomposable jobs,  $a_j$  jobs of type j (for each j) can be allocated only if there exist non-negative  $\{z^l_{j,s} : (l,j) \in [L] \times [N]\}$  satisfying

(1) and 
$$a_{j,s} = a_j$$
 for all  $j, s$ . (2)

Intuitively, the conditions imply that required skill-hours for the set of jobs is less than the available skill-hours of agents. The  $\{z_{j,s}^l\}$  capture a possible way of dividing tasks across multiple category of agents, as they can be interpreted as the number (possibly fraction) of (j,s)-tasks allocated to l-category agents. Note that conditions (1) and (2) are necessary for allocations of decomposable and non-decomposable jobs respectively. These conditions only imply that there exist possible ways of splitting jobs and tasks across different categories of agents to ensure integral number of tasks (jobs) are allocated in case of decomposable (non-decomposable) jobs.

<sup>&</sup>lt;sup>1</sup>Most of our results can be extended to stationary ergodic processes.

For a system with only flexible jobs, different parts of a task can be allocated to different categories and a category can be allocated parts of tasks. Hence,  $\{z_{j,s}^l\}$  can possibly take any value in  $\mathbb{R}^{LNS}_+$ . Thus flexible and decomposable (non-decomposable) jobs need to satisfy condition (1) (condition (2)) which we refer to as FD (FND).

For inflexible jobs, a necessary condition the allocation must satisfy is that each category gets the same integral number of (j, s)-tasks for all s, j, i.e.,

$$z_{j,s}^l \in \mathbb{Z}_+ \text{ s.t. } z_{j,s}^l = z_{j,s'}^l \text{ for all } s, s', j.$$
 (3)

An allocation of inflexible and decomposable (non-decomposable) jobs needs to satisfy conditions (1) (condition (2)) and (3), which we refer to as *ID* (*IND*).

For simplicity, in this work we focus on systems with only a single one of these four classes of jobs.<sup>2</sup> For brevity we use the same abbreviations to refer to class of job, as we use for the necessary conditions. Thus, we have FD, FND, ID, and IND systems.

In crowd systems, the scaling of number of job and agent types, rate of job arrivals, and number of available agents is as follows:  $\lambda(N) = \sum_{j=1}^N \lambda_j$  scales faster than N, i.e.  $\lambda(N) = \omega(N)$  or  $\lim_{N \to \infty} N/\lambda(N) = 0$  and the number of skills S scale slower than N, i.e. S = o(N). In practice, a job requires at most a constant number of skills d, implying there are  $\Omega(S^d)$  possible job types. On the other hand, the number of skills of an agent d' < d as a job generally requires more diversity than a single agent possesses, implying  $M = \sum_l M^l = O(S^{d'})$ . L = O(1), as it relates to variation in reputation levels and hourly rates, and so M = o(N). Beyond these system scalings seen in practice, we assume  $\lambda_j(N) = \omega(1), \forall j \in [N]$  and  $\sum_{j:r_{j,s}>0} \lambda_j(N) = \Omega(N^c)$  for all  $s \in [S]$ , for some c > 0. In the sequel, we assume these scaling patterns and refer to them as crowd-scaling.

## III. CAPACITY, OUTER REGION, AND CENTRALIZED ALLOCATION

In this section we study the limits of a freelance market with centralized allocation and present a centralized algorithm that achieves the limit. We also discuss a simpler upper bound for the capacity region in terms of first-order statistics of the system. These results on ultimate system limits and ways to achieve them are not only important in their own right, but also serve as benchmarks for later discussion of decentralized schemes that provably achieve nearly the same limits.

To formally characterize the maximal supportable arrival rate of jobs we introduce some more notation. For each  $j \in [N]$ , let  $Q_j(t)$  be the number of unallocated jobs that are in the crowd system *just after* allocation epoch t-1. As defined above,  $A_j(t)$  is the number of jobs of type j that arrive between starts of epochs t-1 and t. Let  $D_j(t)$  be the number of jobs of type j that have been allocated to agents at epoch t; we call a job allocated only when all parts have been allocated. Thus the evolution of the process  $Q_j(t)$  can be written as:

$$Q_j(t+1) = Q_j(t) + A_j(t) - D_j(t). (4)$$

Note that at any epoch t, at most  $Q_j(t) + A_j(t)$  type j jobs can be allocated, as this is the total number of type j jobs at that time and hence  $D_j(t) \le Q_j(t) + A_j(t)$ , implying  $Q_j(t) \ge 0$ .

**Notation and Convention.** We denote the interior and the closure of a set C by  $\mathring{C}$  and  $\bar{C}$ , respectively. When we say  $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_N) \in \Lambda \subset \mathbb{R}^{NS}_+$  we mean  $\lambda^S = ((\lambda_1, \lambda_1, \dots, S \text{ times}), (\lambda_2, \lambda_2, \dots, S \text{ times}), \dots) \in \Lambda$ . Also, whenever we say  $\Lambda \subseteq (\supseteq)\Lambda'$  for  $\Lambda' \subset \mathbb{R}^N_+$ , we mean for any  $\lambda \in \mathbb{R}^N_+$ ,  $\lambda \in \Lambda' \Leftarrow (\Rightarrow)\lambda^S \in \Lambda$ .

**Definition 3.** An arrival rate  $\lambda$  is stabilizable if there is a job allocation policy  $\mathcal{P}$  under which  $\mathbf{Q}(t) = (Q_j(t), j \in [N])$  has a finite expectation, i.e.,  $\limsup_{t\to\infty} \mathbb{E}[Q_j(t)] < \infty$ , for all j. The crowd system is called stable under this policy.

**Definition 4.**  $\mathcal{C}_{\Gamma}$ , a closed subset of  $\mathbb{R}^{N}_{+}$  is the capacity region of a crowd system for a given distribution  $\Gamma$  of the agent-availability process if any  $\lambda \in \mathring{\mathcal{C}}_{\Gamma}$  is stabilizable and any  $\lambda \notin \mathcal{C}_{\Gamma}$  is not stabilizable.

<sup>&</sup>lt;sup>2</sup>Extension to combinations of multiple classes is not much different but requires more notation.

# A. Capacity Region and Outer Region

Let us characterize the capacity regions of different classes of crowd systems. For any given set of available agents  $\mathbf{u} = \left(u_i^l : 1 \leq i \leq M^l, 1 \leq l \leq L\right)$ , we define the set of different types of tasks  $(a_{j,s})$  that can be allocated in a crowd system. Note that the necessary conditions to be satisfied for tasks to be allocated are specific to the class of crowd system.

Using the explicit conditions (1), (2), and (3) for tasks (jobs) to be allocated, we define  $C^{\rm FD}(\mathbf{u})$ ,  $C^{\rm FND}(\mathbf{u})$ ,  $C^{\rm ID}(\mathbf{u})$ , and  $C^{\rm IND}(\mathbf{u})$  as the set of tasks that can be allocated in FD, FND, ID, and IND systems respectively for given availability  $\mathbf{u}$ . We denote these sets generically by  $C(\mathbf{u})$  and refer to conditions FD, IFD, FND, and IND generically as *crowd allocation constraint* or *CAC*.

$$C(\mathbf{u}) := \left\{ (a_{j,s} \in \mathbb{Z}_+) : \exists \left( z_{j,s}^l \right) \text{ satisfying CAC} \right\},$$

and  $C(\mathbf{u})$  is the convex hull of  $C(\mathbf{u})$ .

The following theorem generically characterizes capacity regions of different crowd systems.

**Theorem 5.** Given a distribution  $\Gamma$  of agent-availability, i.e.,  $\Gamma(\mathbf{u}) = \Pr(\mathbf{U}(t) = \mathbf{u})$ , for a  $\lambda \notin \bar{\mathcal{C}}(\Gamma)$  there exists no policy under which the crowd system is stable, where

$$\mathcal{C}(\Gamma) = \left\{ \boldsymbol{\lambda} = \sum_{\mathbf{u} \in \mathbb{Z}_+^M} \Gamma(\mathbf{u}) \boldsymbol{\lambda}(\mathbf{u}) : M = \sum_l M^l, \boldsymbol{\lambda}(\mathbf{u}) \in \mathcal{C}(\mathbf{u}) \right\}.$$

For FD, FND, and IND systems, for any  $\lambda \in \mathring{\mathcal{C}}(\Gamma)$  there exists a policy such that the crowd system is stable.

Proof: See Appendix II.A.

This implies that for FD, FND, and IND systems capacity region  $\mathcal{C}_{\Gamma} = \bar{\mathcal{C}}(\Gamma)$  and for ID systems  $\mathcal{C}_{\Gamma} \subseteq \bar{\mathcal{C}}(\Gamma)$  (possibly strict). Note that the conditions FD, FND, ID, and IND (generically CAC) are necessary conditions for a valid allocation. The above theorem implies these conditions are also sufficient, except for ID systems. In Sec. V, we present an alternate characterization of the capacity region for inflexible systems.

Note  $C(\Gamma)$  depends on the distribution of agent availability  $\Gamma$ , but it is hard to obtain this distribution for large and quickly-evolving systems in practice. Hence, a characterization in terms of simpler system statistics is of use. Below is a characterization of a region beyond which no arrival rate can be stabilized. Borrowing terminology from multiterminal Shannon theory, we call this the *outer region*.

For any set  $J \subset [N]$ , define  $\mathcal{N}(J) = \{l \in [L]: \exists j \in J \text{ s.t. } (j,l) \in E\}$  and the closed subset of  $\mathbb{R}^N_+$ ,

$$\mathcal{C}^{out}_{m{\mu}} = \left\{ m{\lambda} : orall J \subset [N], orall s, \sum_{j \in J} \lambda_j r_{j,s} \leq \sum_{l \in \mathcal{N}(J)} \sum_{i \in M^l} \mu^l_i h^l_{i,s} 
ight\}.$$

**Theorem 6.** For any distribution  $\Gamma$  with mean  $\mu$ ,  $\mathcal{C}_{\Gamma} \subseteq \mathcal{C}_{\mu}^{out}$ .

*Proof:* See Appendix II.B.

In general,  $C_{\Gamma}$  is a strict subset of  $C_{\mu}^{out}$  because  $C_{\mu}^{out}$  only captures the balance of skill-hours in the crowd-system, i.e. average skill-hours requirement is no more than average availability, but partial allocation of a task is not acceptable in a crowd system. Moreover, for non-decomposable jobs all tasks of a job have to be allocated simultaneously. Hence, meeting an average skill-hour balance criterion may be far from being sufficient for stability. For inflexible systems the requirements are even stricter, which is likely to increase the gap between the outer region and the true capacity region. In Sec. V we present a tighter outer region for inflexible systems.

In certain scenarios  $\mathcal{C}^{out}_{\mu}$  may be non-empty when  $\mathcal{C}_{\Gamma}$  is empty. For example, consider a simple non-decomposable crowd system with N=L=1 and  $M^1=S=2$ . Let each job require 1 hour of both skills, type i agents have only 1 hour available for skill i and none for other skills,  $\mathbf{U}^1(t)$  be uniformly distributed on  $\{(0,10),(10,0)\}$ , and  $\boldsymbol{\lambda}=(4,4)$ . Then clearly  $\boldsymbol{\lambda}\in\mathcal{C}^{out}_{\mu}$ , but note that at any time there is only one type of skill available, hence no job can be allocated. This implies  $\mathcal{C}_{\Gamma}=\emptyset$ .

## B. Centralized Allocation

end for

Though there exists a policy for each  $\lambda \in \mathring{\mathcal{C}}_{\Gamma}$  that stabilizes the system, these policies may differ based on  $\lambda$  and may depend on the job-arrival and agent availability statistics. Changing policies based on arrival rate and statistics is not desirable in practical crowd systems due to the significant overhead. Below we describe a centralized statistics-agnostic allocation policy which stabilizes any  $\lambda \in \mathring{\mathcal{C}}_{\Gamma}$ . Later we discuss computational cost of this policy for different classes of crowdsourcing system and present simpler distributed (or almost distributed) schemes with provable performance guarantees under some mild assumptions.

To describe the scheme we introduce some more notation. Let  $Q_{j,s}(t)$  be the number of s-tasks (skill s) of type j jobs just after the allocation epoch t-1 and let  $D_{j,s}(t)$  be the number of s-tasks (skill s) of type j jobs allocated at epoch t. Then

$$Q_{j,s}(t+1) = Q_{j,s}(t) + A_j(t) \ 1 (r_{j,s} > 0) - D_{j,s}(t).$$

Note that for all t, due to the CAC condition on allocation,  $D_{j,s}(t) \in C(\mathbf{U}(t))$ . Moreover, there is an additional restriction that  $D_{j,s}(t) \leq Q_{j,s}(t) + A_j(t)$ , as there are  $Q_{j,s}(t) + A_j(t)$  part s of job type j in the system at that time, which in turn implies  $Q_{j,s}(t) \in \mathbb{Z}_+$  for all t. Note that as  $D_{j,s}(t) \in C(\mathbf{U}(t))$ , for non-decomposable systems  $Q_{j,s} = Q_{j,s'}$  for all j, s, s', whereas for decomposable systems they may differ.

```
Algorithm 1 MaxWeight Task Allocation (MWTA)
```

```
Input: \{Q_{j,s}(t): j \in [N], s \in [S]\}, \mathbf{A}(t) and \mathbf{U}(t) at t Output: Allocation of jobs to agents \mathbf{MaxWeight}
```

$$\begin{split} \left(\hat{z}_{j,s}^l(t):l,j,s\right) &= \arg\max_{\left(z_{j,s}^l \in \mathbb{Z}_+:l,j,s\right)} \sum_{j,s} Q_{j,s}(t) \Delta_{j,s} \\ \text{s.t.} \left(z_{j,s}^l\right) \text{ satisfy CAC with } a_{j,s} &= \Delta_{j,s}(t) \forall j,s. \end{split}$$

```
Task Allocation
for j = 1 : N do
   Order j-type jobs arbitrarily, O_i
   for s = 1 : S do
       Use order O_i among non-zero (j, s)-tasks
       while l \leq L and \sum_{k=1}^{l-1} z_{j,s}^k < Q_{j,s}(t) + A_j(t) do Allocate [\sum_{k=1}^{l-1} z_{j,s}^k : \sum_{k=1}^l z_k^l] (j,s) tasks to category l. Here tasks [x:x+y] are task set I=\{\lceil x\rceil, \cdots \lfloor x+y\rfloor\} (in the ordering O_j), (\lceil x\rceil-x) fraction of task \lceil x\rceil and 1+x+y-\lceil x+y\rceil fraction
           of task [x+y].
           l \leftarrow l + 1
       end while
   end for
end for
for l = 1 : L do
   Order agents of category l arbitrarily
   for s = 1 : S do
       Agents pick maximum (as per availability constraint) tasks (or part) in order from \sum_{i} \min(z_{j,s}^{l}, Q_{j,s}(t) +
       A_j(t))r_{j,s} hours
   end for
```

We propose the MaxWeight Task Allocation (MWTA) policy, Alg. 1, to allocate tasks to agents at epoch t based only on the knowledge of  $\mathbf{Q}(t)$ ,  $\mathbf{A}(t)$ , and  $\mathbf{U}(t)$ , and therefore statistics-agnostic. It is based on MaxWeight matching [19], [20].

It is apparent that the MaxWeight part of the algorithm finds a  $\{z_{j,s}^l\}$  that satisfies CAC. The following theorem implies that MWTA allocates tasks optimally. The proof of the theorem is based on adapting the proof of optimality

of the MaxWeight algorithm under the constraints and assumptions of crowd systems. It implicitly relies on the following result.

**Proposition 7.** For any  $\mathbf{u}$  and  $\mathbf{Q}$ , and  $\{z_{j,s}^l\}$  satisfying CAC, the Task Allocation part of MWTA (Alg. 1) gives a feasible allocation for FD, FND, and IND systems.

**Theorem 8.** MWTA (Alg. 1) stabilizes FD, FND, or IND crowd systems for any arrival rate  $\lambda \in \mathring{\mathcal{C}}_{\Gamma}$  (for respective  $\mathcal{C}_{\Gamma}$ ).

# IV. SINGLE-CATEGORY SYSTEMS AND DECENTRALIZED ALLOCATIONS

There is effectively a single category of agents in many platforms with a large population of new freelancers, whose reputations are based on evaluation tests for skills and who are paid at a fixed rate. Hence designing efficient allocation schemes for single-category systems are of particular interest, as this population of agents are significant in ever-evolving crowd systems. Insights drawn from single-category systems are also useful in controlling multicategory systems, Sec. V.

For a single category system (L=1), note that  $z_{j,s}^1=a_{j,s}\in\mathbb{Z}_+$  and hence the feasibility condition (1) reduces to:

$$\sum_{j} a_{j,s} r_{j,s} \leq \sum_{i} u_{i} h_{i,s} \text{ for all } s \in [S], a_{j,s} \in \mathbb{Z}_{+},$$

with condition (2) additionally requiring  $a_{j,s} = a_{j,s'}$  for all j, s, s'. Thus,  $C(\mathbf{u})$  is the set of  $\{a_{j,s}\}$  satisfying the above conditions for respective classes of jobs and  $C(\Gamma)$  is the weighted (by  $\Gamma(\mathbf{u})$ ) sum of convex hulls of  $C(\mathbf{u})$ s, here  $C_{\Gamma} = \bar{C}(\Gamma)$ .

 $\mathcal{C}^{out}_{\boldsymbol{\mu}} \text{ has a simple characterization as well. As for any } j \in [N], \ (j,1) \in E, \text{ and } \mathcal{N}(J) = 1 \text{ for all } J \subset [N], \\ \sum_{l \in \mathcal{N}(J)} \sum_{i \in [M^l]} \mu_i^l h_{i,s}^l = \sum_{i \in [M]} \mu_i h_{i,s}. \text{ Thus it is sufficient to satisfy the inequality for } J = [N], \text{ and hence, } \\ \mathcal{C}^{out}_{\boldsymbol{\mu}} = \Big\{ \boldsymbol{\lambda} : \sum_{j \in [N]} \lambda_j r_j \leq \sum_{i \in [M]} \mu_i h_i \Big\}.$ 

The MaxWeight computation in MWTA for single-category systems turns out to be the following integer linear program (ILP), which is related to knapsack problems.

$$\arg \max_{\{\Delta_{j,s}:j,s\}} \sum_{j,s} Q_{j,s} \Delta_{j,s}$$
s.t. 
$$\sum_{j} \Delta_{j,s} r_{j,s} \leq \sum_{i} u_{i} h_{i,s} \forall s \in [S],$$

$$\Delta_{j,s} = \Delta_{j,s'}, \forall s, s', j \text{ (only for ND)}$$

For decomposable and non-decomposable systems, this is a single knapsack and multi-dimensional knapsack problem [34], respectively, and hence NP-hard. There exist fully polynomial time approximations (FPTAS) for single knapsack, whereas for multi-dimensional knapsack only polynomial time approximations (PTAS) are possible [34]. With this approximation, say  $1 - \epsilon$ , the MWTA policy stabilizes  $(1 - \epsilon)\mathring{\mathcal{C}}_{\Gamma} = \{\lambda : \frac{\lambda}{1 - \epsilon} \in \mathring{\mathcal{C}}_{\Gamma}\}$ . Also, note that for large crowd systems each  $\lambda_i$  is large and hence stabilizing any  $\lambda$  with  $\lambda + 1 \in \mathring{\mathcal{C}}_{\Gamma}$  is almost optimal. The above ILP can be relaxed to obtain a linear program, an allocation based on which achieves this approximation (see Appendix I.A for details).

### A. Decentralized Allocations

Now we show that due to the structure of the crowd allocation problem and the fact that crowd systems are large, simple allocation schemes with minimal centralized control achieve good performance under mild assumptions on arrival and availability processes. Interestingly, though the centralized optimal allocation requires solving a knapsack problem at each epoch and greedy schemes are known to be sub-optimal for knapsack problems [34], we propose two simple greedy schemes that are almost optimal with good performance guarantees. One of them, called

GreedyAgent allocation provably performs well for decomposable systems and offers the freedom of selection to freelancers. Another, called GreedyJob allocation has provable performance guarantees for both decomposable and non-decomposable systems while allowing customers (job requesters) the freedom of selection. Thus, in some sense, this shows that though greedy algorithms can be suboptimal for an arbitrary allocation instant (at each epoch), for a dynamical system over long time, its performance is good.

# Algorithm 2 GreedyAgent Allocation

```
Input: \mathbf{A}(t)
Output: Job to agent allocations
\mathcal{A}: set of agents, \mathcal{T}: set of tasks
while A and T non-empty do
  Agents in A contend (pick random numbers) and a wins
  for each skill with non-zero skill hour do
     a picks as many integral tasks as it can pick
     if a has remaining available hour then
        a Picks from remaining parts of the partially allocated task
       if a has remaining available hour then
           a picks part of any unallocated task
       end if
     end if
     Remove fully allocated tasks from \mathcal{T}
  end for
  \mathcal{A} = \mathcal{A} \setminus \{a\}
end while
Tasks with partial allocations are not allocated
```

In GreedyAgent allocation (Alg. 2), agents themselves figure out the allocation via contention, without any central control. Agents need no knowledge about the agent population, but do need information on the available pool of jobs and have to agree on certain norms. In most freelance market platforms, this information is readily available, and so an algorithm like this is natural. As expected, this scheme may not be able to stabilize any arrival rate in  $\mathring{\mathcal{C}}_{\Gamma}$  for any ergodic job-arrival and agent-availability processes, but it has good theoretical guarantees under some mild assumptions on the job arrival and agent availability processes.

**Definition 9.** A random variable 
$$X$$
 is Gaussian-dominated if  $\mathbb{E}[X^2] \leq \mathbb{E}[X]^2 + \mathbb{E}[X]$  and for all  $\theta \in \mathbb{R}$ ,  $\mathbb{E}[e^{\theta(X-\mathbb{E}[X])}] \leq \exp\left\{\frac{((\mathbb{E}[X^2]-\mathbb{E}[X]^2)\theta^2}{2}\right\}$ 

**Definition 10.** A random variable X is Poisson-dominated if for all  $\theta \in \mathbb{R}$ ,  $\mathbb{E}[e^{\theta(X-\mathbb{E}[X])}] \leq e^{\mathbb{E}[X](e^{\theta}-\theta-1)}$ .

Note that these domination definitions imply that the variation of the random variable around its mean is dominated in a moment generating function sense by that of a Gaussian or Poisson random variable.<sup>3</sup> Such a property is satisfied by many distributions including Poisson and binomial that are used to model arrival processes for many systems, e.g., telephone networks, internet, call centers, and some freelance markets [5], [6]. It is not hard to show that sub-Gaussian distributions (standard in machine learning [36]) that are symmetric around their mean, are Gaussian-dominated.

The following theorem gives a guarantee on the performance of GreedyAgent, under mild restrictions on the jobarrival and agent-availability processes. Independence assumptions are not too restrictive for large crowd systems, where jobs and agents may come from different well-separated geographies or organizational structures.

**Theorem 11.** If the arrival processes  $\{A_j(t)\}$  and the agent availability processes  $\{U_i(t)\}$  are i.i.d. across time and independent across types (jobs and agents) and all these processes are Gaussian-dominated (and/or Poisson-dominated), then for any given  $\alpha \in (0,1]$ , there exists an  $N_\alpha$  such that GreedyAgent allocation stabilizes any

<sup>&</sup>lt;sup>3</sup>Domination in this sense is used in bandit problems [35].

arrival rate  $\lambda \in (1-\alpha)\mathcal{C}^{out}_{\mu} := \left\{\lambda : \frac{1}{1-\alpha}\lambda \in \mathcal{C}^{out}_{\mu}\right\}$  for any single-category decomposable crowd system with  $N \geq N_{\alpha}$ . Moreover, for any arrival rate in  $(1-\alpha)\mathcal{C}^{out}_{\mu}$ , at steady state, after an allocation epoch, the number of unallocated tasks is  $O(S \log N)$  with probability  $1-o\left(\frac{1}{N^2}\right)$ .

```
Proof: See Appendix II.E.
```

As  $\mathcal{C}_{\Gamma} \subseteq \mathcal{C}_{\mu}^{out}$ , this implies that the greedy scheme stabilizes an arbitrarily large fraction of the capacity region, under the assumptions on the arrival and availability processes. As S = o(N), more specifically  $O(N^c)$  for c < 1, the above bound on number of jobs imply that there are o(N) unallocated tasks at any time. This in turn implies that unallocated tasks per type (average across types) is o(1), i.e., vanishingly small number of tasks per type are unallocated as the system scales.

In GreedyAgent, there is no coordination among agents while picking tasks within jobs. Hence in a non-decomposable system, many tasks may be picked by agents but only few complete jobs are allocated. As more and more jobs accumulate, the chance of this happening increases, resulting in more accumulation. This can result in the number of accumulated jobs growing without bound, as formalized below.

**Proposition 12.** There exists a class of non-decomposable crowd systems with Poisson-dominated (as well as Gaussian-dominated) distributions of arrival and availability, such that the system is not stable under GreedyAgent allocation.

```
Proof: See Appendix II.F.
```

Hence, we propose another simple greedy scheme that works for both decomposable and non-decomposable systems. The GreedyJob allocation scheme (Alg. 3) is completely distributed and hence a good fit for crowd systems. GreedyJob has similar performance guarantees for both decomposable and non-decomposable systems as GreedyAgent has for decomposable systems only.

# Algorithm 3 GreedyJob Allocation

```
Input: \mathbf{U}(t)
Output: Job to agent allocations \mathcal{J}: set of all jobs
while Available skill-hours of agents and \mathcal{J} \neq \emptyset do
Jobs in \mathcal{J} contend (pick random numbers) and J wins
if J finds agents to allocate all tasks then
Allocate to those agents
else
J does not allocate anything
end if
\mathcal{J} = \mathcal{J} \setminus \{J\}
end while
```

**Theorem 13.** If the arrival processes  $\{A_j(t)\}$  and the agent availability processes  $\{U_i(t)\}$  are i.i.d. across time and independent across types (jobs and agents), all these processes are Gaussian-dominated (and/or Poisson-dominated) and  $\forall s, s', |\sum_i \mu_i h_{i,s} - \sum_i \mu_i h_{i,s'}|$  is O(subpoly(N)), then for any given  $\alpha \in (0,1]$ ,  $\exists N_\alpha$  such that GreedyJob allocation stabilizes any arrival rate  $\lambda \in (1-\alpha)\mathcal{C}^{out}_{\mu} := \left\{\lambda : \frac{1}{1-\alpha}\lambda \in \mathcal{C}^{out}_{\mu}\right\}$  for any single-category crowd-system with  $N \geq N_\alpha$ . Moreover, for any arrival rate in  $(1-\alpha)\mathcal{C}^{out}_{\mu}$ , at steady state, after an allocation epoch, total number of unallocated jobs (adding all types) is  $O(\log N)$  with probability  $1-o\left(\frac{1}{N^2}\right)$ .

```
Proof: See Appendix II.G.
```

In Sec. V we propose a decentralized scheme for multi-category systems that uses the two single-category decentralized schemes as building blocks. At the end of Sec. V we briefly discuss the suitability of these decentralized schemes for crowd systems in terms of implementability on crowd platforms.

#### V. Multi-Category Systems

Sec. III characterized the capacity region and developed an optimal centralized scheme for crowd systems in, whereas Sec. IV discussed simple decentralized schemes for single-category systems. Here we return to multicategory systems, briefly discussing computational aspects of MWTA, followed by an alternate approach to the capacity and outer region of inflexible systems that yields a simple optimal scheme. We also present a decentralized scheme based on insights from the optimal scheme and the decentralized allocations in Sec. IV.

The MWTA scheme, which is throughput optimal for FD, FND, and IND systems, involves solving an NP-hard problem for multi-category systems. For multi-category systems this is from the general class of packing integer programs, for which constant factor approximation algorithms exist under different assumptions on the problem parameters [37]. These assumptions do not generally hold for MaxWeight allocation under the CAC constraint. Rather, we follow the same steps of LP relaxation and obtain a scheme that stabilizes any  $\lambda$  for  $\lambda + 1 \in \mathring{\mathcal{C}}_{\Gamma}$ , since for large systems this is better than any arbitrarily close approximation scheme (as  $\lambda \to \infty$  as  $N \to \infty$ ).

# A. Inflexible System

Below we present a characterization of the capacity region of inflexible systems in terms of the bipartite graph G = (V, E), which captures the restriction of job-agent allocations.

**Theorem 14.** Any  $\lambda$  can be stabilized if  $\lambda \in \mathring{C}^I$ , where

$$\mathcal{C}^I = \left\{ oldsymbol{\lambda} = \sum_{l \in [L]} oldsymbol{\lambda}^{(l)}: \ oldsymbol{\lambda}^{(l)} \in \mathcal{C}^{(l)}_{\Gamma_l}, \lambda^l_j = 0 \ \ \textit{for all} \ (j,l) 
otin E 
ight\},$$

where  $C_{\Gamma_l}^{(l)}$  is the capacity region of a single category system with an agent availability distribution  $\Gamma_l = \Gamma\left(U_i^l: i \in [M^l]\right)$ . Moreover, no  $\lambda \notin \bar{\mathcal{C}}^I$  can be stabilized, i.e., for inflexible systems the capacity region  $\mathcal{C}_{\Gamma} = \bar{\mathcal{C}}^I$ .

Proof: See Appendix II.H.

This theorem has the following simple consequence. Consider separate pools of agents for each different category, cf. [7], which has agent-availability distributions  $\{\Gamma_l: l \in [L]\}$ . Each such pool (category) of agents l can stabilize job-arrival rates in  $\mathring{\mathcal{C}}_{\Gamma_l}^{(l)}$ . Thus if the job arrival process of each job type j can be split in such a way that pool (category) l of agents sees an arrival rate  $\lambda_j^l$ , where  $\lambda_j^l > 0$  only if  $(j,l) \in E$ , while ensuring that  $\{\lambda_j^l: j\} \in \mathring{\mathcal{C}}_{\Gamma_l}^{(l)}$ , the system would be stable.

In a server farm where jobs can be placed on any of the server queues, the join-shortest-queue (JSQ) policy stabilizes any stabilizable rate [20]. JSQ gives an arriving job to the server with the shortest queue and each server serves jobs in FIFO order. For multi-category crowd systems, we can draw a parallel between servers and agent pools. In addition we have constraints on job placement given by G and also have to do allocations of jobs among the agents in the pool optimally (unlike JSQ we do not have FIFO/LIFO specified). Thus we have to adapt JSQ appropriately based on our insights about optimal operation of crowd systems.

We propose a statistics-agnostic scheme, JLTT-MWTA (Alg. 4) that has two parts: JLTT (join least total task) directs arrivals to appropriate pools of agents and MWTA allocates jobs in each pool separately. Letting  $Q_{j,s}^l(t)$  be the number of unallocated (j,s)-tasks in lth pool just after epoch t-1, JLTT uses these quantities to direct jobs to appropriate pools whereas MWTA uses them to allocate tasks within each pool.

The JLTT part is computationally light. The central controller only needs to know  $\mathbf{Q}(t)$  and has to pick the minimally loaded  $(\min_l \sum_s Q_{j,s}^l)$  pools of agents to direct jobs (type j). To perform MWTA in each pool, a PTAS, FPTAS, or LP relaxation scheme can be used.

Unlike JSQ, where service discipline in each server is fixed and the goal is to place the jobs optimally, we have jobs with multi-dimensional service requirements from time-varying stochastic servers (agent-availability) and have to place jobs as well as discipline the service in each random and time-varying virtual pool. Thus optimality of JSQ cannot be claimed in our case. But as stated below, JLTT division followed by MWTA allocation is indeed optimal.

**Theorem 15.** JLTT-MWTA stabilizes any  $\lambda \in \mathring{\mathcal{C}}^I$ .

Proof: See Appendix II.I.

# Algorithm 4 JLTT-MWTA: Divide and Allocate

```
Input: \mathbf{A}(t), \ \mathbf{U}(t), \ \mathbf{Q}(t)
Output: Job division and allocation
Create pool l with category l agents (\forall l \in [L])
JLTT: Join Least Total Task
for each (j,s) do

Count number of unallocated (j,s)-tasks in pool l: Q_{j,s}^l(t)
Divide A_j(t)1(r_{j,s}>0) tasks equally among pools \arg\min_{l:(j,l)\in E}\sum_s Q_{j,s}^l(t)
end for
In each pool l run MWTA for single-category system
```

An important aspect of JLTT-MWTA is that job allocations within each pool can happen independently of each other. The central controller only has to make a decision on how to split the jobs based on the current system state information. This allows a more distributed allocation along the lines of following hierarchical organizational structure [38]. First, the central controller divides jobs for different agent-pools based on  $\{Q_{j,s}^l\}$ . Then in each agent pool, allocations are according to GreedyJob allocation, which works for both decomposable and non-decomposable single-category systems. The distributed scheme that we propose here is an improvisation of the above JLTT scheme followed by GreedyJob allocation in each pool. We call it Improvised JLTT and GreedyJob Allocation (Alg. 5).

# Algorithm 5 Improvised JLTT and GreedyJob Allocation

```
Input: \mathbf{A}(t), \ \mathbf{U}(t), \ \mathbf{Q}(t)
Output: Job division and allocation

Improvised JSQ for each job-type j and each skill s:

n=0
N_j^l=Q_j^l(t)
While n < A_j(t)
Send 1 task to the category l^* with lowest index among \arg\min_{l:(j,l) \in E} N_j^l
Increase N_j^{l^*} and n each by 1

End While

Allocations within each pool l:
Run GreedyJob allocation
```

First note that unlike JLTT-MWTA, here we only maintain number of unallocated jobs and do not maintain number of unallocated tasks for each skill s. This is because as GreedyJob allocation is used as allocation scheme in each pool,  $Q_{j,s}^l = Q_{j,s'}^l$  for all s, s'.

This algorithm is also simple to implement. The central controller only needs to track the number of unallocated jobs  $(Q_j^l(t))$  from the previous epoch and set  $N_j^l = Q_j^l$ . For any arriving job of type j, the central controller sends the job to the pool with minimum  $N_j^l$  and updates  $N_j^l$ . This continues until the next epoch, when the  $N_j^l$  are reset to new  $Q_j^l$  values.

Recall that Sec. I gave a simple example of a fully distributed scheme where jobs pick agents greedily (from the set of feasible agents as per G) and showed it was not a good scheme. Improvisation of JLTT is proposed for a better performance guarantee, while GreedyJob in each pool is proposed for implementability and freedom of selection for customers. It is not hard to prove Improvised JLTT followed by MWTA is optimal for any arrival and availability process satisfying the assumptions of Sec. II. Below we present performance guarantee for Improvised JLTT and GreedyJob allocation.

To present performance guarantees of the distributed scheme we give an outer region  $\mathcal{C}^O$  for the system, along the lines of the alternative characterization  $\mathcal{C}^I$  of the capacity region  $\mathcal{C}_{\Gamma}$  for inflexible systems.

**Theorem 16.** Inflexible crowd systems cannot be stabilized for  $\lambda \notin C^O$ , where  $C^O = \left\{ \lambda : \lambda = \sum_{l \in [L]} \lambda^{(l)} \text{ where } \lambda^{(l)} \in C^{out}_{\mu^l} \right\}$  and  $C^{out}_{\mu^l}$  is the outer region for the single category system comprising the lth category (pool) of agents with

$$oldsymbol{\mu}^l = \mathbb{E}\left[\mathbf{U}^l\right].$$

Proof: See Appendix II.J.

For job allocation in server farms, extant performance guarantees are mostly for symmetric load, i.e., symmetric (almost) service and job arrival rates and regular graphs, cf. [30], [31]. Unlike server farms, symmetric load (in terms of skill-hours) is not guaranteed in crowd systems by symmetric arrival rates and graphs. This is because different types of jobs have different skill and hour requirements. The following guarantee for crowd systems is for bounded asymmetry (sub-polynomial variation) in agent availability, complete graph, asymmetric job arrival rates, and asymmetric job requirements (extendable to regular graphs with additional assumptions on symmetry of job arrival rates and requirements). Note that because of the inflexibility constraint, a multi-category system with a complete graph is not equivalent to a single-category system.

**Theorem 17.** Without loss of generality assume the same ordering of agent types in each category, i.e.,  $M^l = M^{l'} = M/L$  and  $h^l_i = h^{l'}_i$  for all l, l', i. If the arrival processes  $\{A_j(t)\}$  and the agent availability processes  $\{U^l_i(t)\}$  are i.i.d. across time and independent across types (jobs and agents), all these processes are Gaussian-dominated (and/or Poisson-dominated),  $\sum_i \max_{l,l'} |\mu^l_i - \mu^{l'}_i|$  and  $\max_{l,s,s'} |\sum_i \mu^l_i(h^l_{i,s} - h^l_{i,s'})|$  are O(subpoly(N)) and G is complete bipartite, then for any given  $\alpha \in (0,1]$ ,  $\exists N_\alpha$  such that Improvised JLTT and Greedy-job stabilizes any  $\lambda \in (1-\alpha)\mathcal{C}^O := \left\{\lambda : \frac{1}{1-\alpha}\lambda \in \mathcal{C}^O\right\}$  and the maximum number of unallocated jobs (across all types) is  $O(\log N)$  with probability  $1-o\left(\frac{1}{N^2}\right)$ .

Proof: See Appendix II.K.

The proofs of Thm. 11, 13, and 17 are all based on constructing queue-processes (different for the algorithms) that stochastically dominate the number of unallocated jobs, and bounding the steady state distributions of these processes using Loynes' construction and moment generating function techniques.

# B. Implementation of Decentralized Schemes on Crowd Platforms

We have described the allocation schemes at the level of system abstraction and discussed their performance. These schemes can be easily implemented on crowd platforms as well.

GreedyAgent allocation is completely decentralized, only requiring agents to abide by a norm for picking partial tasks, which can be enforced by randomized vigilance and penalizing norm violators in reputation. If the payments are the same, as it generally is in single-category systems where all jobs require same quality, there is no incentive for agents to deviate from the norm.

Any arbitrary contention method among agents will work for the algorithm, and hence the crowdsourcing platform only needs to ensure that no two allocations are done simultaneously (as practiced in airlines booking). Multiple allocations can also be allowed by the platform if they do not conflict. Here the platform has to ensure that an agent can place requests only for an amount of tasks it can actually perform, given the constraints on available hours. Also, only one agent can request for a task or a certain part of it. Once the agent has been declined, it can place request(s) for task(s) of the same or lesser hours. This can either be enforced by appropriate modification of the portals by keeping tracks of total hours of requests placed or by vigilance. GreedyJob can also be easily implemented on a crowd platform. The platform has to ensure that jobs request agents and not the other way around. One way to implement this is to allow jobs to place requests for agents while ensuring they do not request more than the required service. Also, the platform has to ensure that skill-hour requests of no two jobs collide. This again can be ensured by serializing the requests as above. Agents are expected to accept a requested task, as there is no difference between tasks involving same skill since payments are the same. This can also be ensured by linking agent rating to rate of task-request acceptance.

It is apparent that GreedyJob offers choice to customers and GreedyAgent offers choice to agents. By allowing a customer (or an agent) to decline an approaching agent (or a customer) request and to explore more options, only one option at a time, the platform can provide freedom of choice to agents and customers under both schemes while operating at capacity.

In case of multi-category systems, the platform only needs to direct arriving jobs to the appropriate pool of agents, based on current backlog; the rest of the allocation happens as per GreedyJob. Directing a job to a category of agents can be implemented in a crowd platform by making the job visible only to freelancers of that category and

vice versa (similar to filtering done by search engines and online social networks) or through explicit hierarchical organization into pools [38].

#### VI. BEYOND THE CAPACITY REGION

We have now characterized the capacity (and outer) regions of different classes of crowd systems, shown the existence of computationally feasible centralized schemes that achieve these regions, and presented simple distributed schemes with minimal centralized intervention and good performance guarantees for any arrival rate within the capacity region. In crowd systems, however, arrival rates may not be within the capacity region, since the platform may have little or no control on resource (freelancer) planning, unlike traditional communication networks or cloud computing systems. Hence, an important aspect of crowd systems is to turn down job requests. Indeed, deciding to decline a job must be done as soon as the job arrives because dropping a job after first being accepted adversely affects the reputation of the crowd platform.

Here, we propose a centralized scheme for a crowd system to decline jobs on arrival in a way that is fair across all job types. Our scheme is statistics-agnostic and works even for independent but non-stationary arrival and availability processes.

We solve the following problem: given an arrival rate  $\lambda$ , design a statistics-agnostic policy to accept  $(1 - \beta)\lambda$  jobs on average and allocate them appropriately such that  $\beta$  is the minimum for which the crowd system is stable. Note that if  $\lambda \in \mathring{\mathcal{C}}_{\Gamma}$ , the minimum  $\beta$  is 0, else, it is strictly positive. We want to design a statistics-agnostic policy without the knowledge of  $\lambda$  and  $\mathcal{C}_{\Gamma}$ . As a benchmark, we consider the following problem for  $\epsilon > 0$ , when  $\lambda$  and  $\mathcal{C}_{\Gamma}$  are known.

$$\min \beta \in [0, 1]$$
 s.t.  $(1 - \beta)\lambda + \epsilon \mathbf{1} \in \mathcal{C}_{\Gamma}$ . (6)

Given  $\beta^*$ , optimum of (6),  $(1 - \beta^*)\lambda$  is within  $\epsilon$  of the optimal rate of accepted jobs for which the system is stabilizable.

As we want a scheme that is agnostic of  $\lambda$  and  $\mathcal{C}_{\Gamma}$ , we propose the following simple scheme, for  $\nu > 0$  and  $\tilde{Q}_{i,s}^l(t)$  is the number of unallocated accepted (j,s)-tasks ((j,s)-tasks directed to category l) in the system.

$$\beta(t) = \arg\min_{\beta \in [0,1]} \\ \begin{cases} \beta \sum_{j} A_{j}(t) - \nu \beta \sum_{j,s:r_{j,s}>0} \tilde{Q}_{j,s}(t) A_{j}(t) & \text{(I)} \\ \beta \sum_{j} A_{j}(t) - \nu \beta \sum_{j,s:r_{j,s}>0} \min_{l} \tilde{Q}_{j,s}^{l}(t) A_{j}(t) & \text{(II)} \end{cases} \\ \text{Job is accepted w.p. } 1 - \beta(t), \text{ accepted jobs } \tilde{\mathbf{A}}(t) & \text{(I \& II)} \end{cases}$$
 For accepted jobs run 
$$\begin{cases} \text{MWTA} & \text{(I)} \\ \text{JLtT-MWTA} & \text{(II)} \end{cases}$$
 (7)

Steps marked by I (II) are applicable for FD, FND and IND (ID and IND) systems.

**Theorem 18.** Crowd system with jobs accepted and allocated according to (7) is stable and  $\sum_{j} \lambda_{j} (1 - \beta^{*}) - \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\sum_{j} \tilde{A}_{j}(t)]$  can be made arbitrarily small for an appropriately chosen  $\nu$ , for all sufficiently large T.

This theorem demonstrates that by following the job acceptance and allocation method (7), the crowd system can be stabilized while ensuring the average number of accepted jobs per allocation epoch is arbitrarily close to the optimal number of accepted jobs per allocation period. Note that as all jobs (across all types) are accepted with the same probability, the above result also implies that  $(1-\beta^*)\lambda_j - \frac{1}{T}\sum_{t=1}^T \mathbb{E}[\tilde{A}_j(t)]$  is small. It can be shown that the above scheme works for time-varying systems  $(\mathbb{E}[\mathbf{A}(t)] = \boldsymbol{\lambda}(t))$  and  $\mathbf{U} \sim \Gamma^t(t)$  as well guaranteeing small  $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^T\sum_j\left(\lambda_j(t)(1-\beta^*(t)) - \mathbb{E}[\tilde{A}_j(t)]\right)$ , where  $\beta^*(t)$  is the solution of (6) for  $\boldsymbol{\lambda}=\boldsymbol{\lambda}(t)$  and  $\mathcal{C}_\Gamma=\mathcal{C}_{\Gamma^t}$ . sectionConclusion

Human information processing, structured through freelance markets, is an emerging structure for performing informational work by capturing the cognitive energy of the crowd. It is important to understand the fundamental limits and optimal designs for such systems.

In this work we provide a characterization of the work capacity of crowd systems and present two statistic-agnostic job allocation schemes MWTA (flexible jobs) and JLTT-MWTA (inflexible jobs) to achieve limits. To ensure low computational load on the crowd platform provider and freedom of choice for job requesters, we present simple decentralized schemes, GreedyAgent, GreedyJob, and Improvised JLTT-GreedyJob that (almost) achieve capacity with certain performance guarantees. These decentralized schemes are easy to implement on crowd platforms, require minimal centralized control, and offer freedom of self-selection to customers: all desirable qualities for any crowd platform. Due to quick evolution and unpredictability of freelancer resources, crowd systems may often operate outside capacity, which inevitably results in huge backlogs. Backlogs hurt the reputation of the platform, and so we also propose a scheme that judiciously accepts or rejects jobs based on the system load. This scheme is fair in accepting jobs across all types and accepts the maximum number of jobs under which the system can be stable.

#### REFERENCES

- [1] A. Chatterjee, L. R. Varshney, and S. Vishwananth, "Work capacity of freelance markets: Fundamental limits and decentralized schemes," in *Proc. 2015 IEEE INFOCOM*, Apr. 2015, pp. 1769–1777.
- [2] D. Tapscott and A. D. Williams, Wikinomics: How Mass Collaboration Changes Everything, expanded ed. New York: Portfolio Penguin, 2006.
- [3] Y. Benkler, *The Wealth of Networks: How Social Production Transforms Markets and Freedom.* New Haven, CT: Yale University Press, 2006.
- [4] D. Bollier, *The Future of Work: What It Means for Individuals, Businesses, Markets and Governments.* Washington, DC: The Aspen Institute, 2011.
- [5] D. F. Bacon, E. Bokelberg, Y. Chen, I. A. Kash, D. C. Parkes, M. Rao, and M. Sridharan, "Software economies," in *Proc. FSE/SDP Workshop Future Softw. Eng. Research (FoSER 2010)*, Nov. 2010, pp. 7–12.
- [6] M. Vukovic and O. Stewart, "Collective intelligence applications in IT services business," in *Proc. IEEE 9th Int. Conf. Services Comput.* (SCC), Jun. 2012, pp. 486–493.
- [7] L. R. Varshney, S. Agarwal, Y.-M. Chee, R. R. Sindhgatta, D. V. Oppenheim, J. Lee, and K. Ratakonda, "Cognitive coordination of global service delivery," arXiv:1406.0215v1 [cs.OH]., Jun. 2014.
- [8] C. Shirky, Cognitive Surplus: Creativity and Generosity in a Connected Age. Penguin, 2010.
- [9] T. W. Malone, R. Laubacher, and C. Dellarocas, "The collective intelligence genome," *MIT Sloan Manage. Rev.*, vol. 51, no. 3, pp. 21–31, Spring 2010.
- [10] K. J. Boudreau and K. R. Lakhani, "Using the crowd as an innovation partner," *Harvard Bus. Rev.*, vol. 91, no. 4, pp. 60–69, Apr. 2013.
- [11] D. DiPalantino and M. Vojnović, "Crowdsourcing and all-pay auctions," in Proc. 10th ACM Conf. Electron. Commer. (EC'09), Jul. 2009, pp. 119–128.
- [12] N. Archak and A. Sundararajan, "Optimal design of crowdsourcing contests," in Proc. Int. Conf. Inf. Syst. (ICIS), 2009, p. 200.
- [13] K. J. Boudreau, N. Lacetera, and K. R. Lakhani, "Incentives and problem uncertainty in innovation contests: An empirical analysis," *Manage. Sci.*, vol. 57, no. 5, pp. 843–863, May 2011.
- [14] G. V. Ranade and L. R. Varshney, "To crowdsource or not to crowdsource?" in *Proc. AAAI Workshop Human Comput. (HCOMP'12)*, Jul. 2012, pp. 150–156.
- [15] D. R. Karger, S. Oh, and D. Shah, "Budget-optimal task allocation for reliable crowdsourcing systems," *Oper. Res.*, vol. 62, no. 1, pp. 1–24, Jan.-Feb. 2014.
- [16] C.-J. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets," in Proc. 26th AAAI Conf. Artif. Intell., Jul. 2012, pp. 45–51.
- [17] D. V. Oppenheim, L. R. Varshney, and Y.-M. Chee, "Work as a service," in *Advanced Web Services*, A. Bouguettaya, Q. Z. Sheng, and F. Daniel, Eds. Springer, 2014, pp. 409–430.
- [18] J. Kleinberg and É. Tardos, Algorithm Design. Addison-Wesley, 2005.
- [19] M. J. Neely, Stochastic Network Optimization with Application to Communication and Queueing Systems. Morgan & Claypool Publishers, 2010.
- [20] R. Srikant and L. Ying, Communication Networks: An Optimization, Control and Stochastic Networks Perspective. Cambridge University Press, 2014.
- [21] M. L. Pinedo, Scheduling: Theory, Algorithms, and Systems. Springer, 2012.
- [22] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.
- [23] —, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. Inf. Theory*, vol. 39, no. 2, pp. 466–478, Mar. 1993.
- [24] A. Eryilmaz, R. Srikant, and J. R. Perkins, "Stable scheduling policies for fading wireless channels," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 411–424, Apr. 2005.
- [25] M. J. Neely and E. Modiano, "Capacity and delay tradeoffs for ad hoc mobile networks," *IEEE Trans. Inf. Theory*, vol. 51, no. 6, pp. 1917–1937, Jun. 2005.
- [26] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *Proc.* 2012 IEEE INFOCOM, Mar. 2012, pp. 702–710.

- [27] I. Menache, A. Ozdaglar, R. Srikant, and D. Acemoglu, "Dynamic online-advertising auctions as stochastic scheduling," in Proc. Workshop Econ. Netw. Syst. Comput. (NetEcon '09), Jul. 2009.
- [28] B. Tan and R. Srikant, "Online advertisement, optimization and stochastic networks," *IEEE Trans. Autom. Control*, vol. 57, no. 11, pp. 2854–2868, Nov. 2012.
- [29] S. Chen, P. Sinha, and N. B. Shroff, "Scheduling heterogeneous delay tolerant tasks in smart grid with renewable energy," in *Proc.* 51st IEEE Conf. Decision Control, Dec. 2012, pp. 1130–1135.
- [30] J. N. Tsitsiklis and K. Xu, "Queueing system topologies with limited flexibility," in Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst., Dec. 2013, pp. 167–178.
- [31] M. Bramson, Y. Lu, and B. Prabhakar, "Randomized load balancing with general service time distributions," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, Jun. 2010, pp. 275–286.
- [32] V. Shah and G. de Veciana, "Performance evaluation and asymptotics for content delivery networks," in *Proc. 2014 IEEE INFOCOM*, Apr.-May 2014, pp. 2607–2615.
- [33] D. Wang, "Computing with unreliable resources: Design, analysis and algorithms," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, Jun. 2014.
- [34] H. Kellerer, U. Pferschy, and D. Pisinger, Knapsack Problems. Springer, 2004.
- [35] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Found. Trends Mach. Learn.*, vol. 5, no. 1, pp. 1–122, Dec. 2012.
- [36] R. Keshavan, A. Montanari, and S. Oh, "Matrix completion from noisy entries," in *Advances in Neural Information Processing Systems* 22. Cambridge, MA: MIT Press, 2009, pp. 952–960.
- [37] C. Chekuri and S. Khanna, "On multidimensional packing problems," SIAM J. Comput., vol. 33, no. 4, pp. 837-851, 2004.
- [38] S. Agarwal, S. Kenkre, V. Pandit, and B. Sengupta, "Studying the evolution of skill profiles in distributed, specialization driven service delivery systems through work orchestration," in *Proc. SRII Global Conf. 2011*, Apr. 2011, pp. 201–213.

## APPENDIX I

# A. Computation for Centralized Allocation

For a single category system (L=1), note that  $z_{j,s}^1=a_{j,s}\in\mathbb{Z}_+$  and hence the feasibility condition (1) becomes

$$\sum_{j} a_{j,s} r_{j,s} \le \sum_{i} u_i h_{i,s} \text{ for all } s \in [S], a_{j,s} \in \mathbb{Z}_+,$$

with condition (2) additionally requiring  $a_{j,s} = a_{j,s'}$  for all j, s, s'. Thus,  $C(\mathbf{u})$  is the set of  $\{a_{j,s}\}$  satisfying the above conditions for respective classes of jobs (as well as systems) and C is the weighted (by  $\Gamma(\mathbf{u})$ ) sum of convex hulls of  $C(\mathbf{u})$ s.

 $\mathcal{C}^{out}_{\mu}$  has a simple characterization as well. As for any  $j \in [N]$ ,  $(j,1) \in E$ , and  $\mathcal{N}(J) = 1$  for all  $J \subset [N]$ ,  $\sum_{l \in \mathcal{N}(J)} \sum_{i \in [M^l]} \mu_i^l h_s^{l,i} = \sum_{i \in [M]} \mu_i h_{i,s}$ . Thus it is sufficient to satisfy the inequality for J = [N], and hence,

$$\mathcal{C}^{out}\boldsymbol{\mu} = \left\{ \boldsymbol{\lambda} : \sum_{j \in [N]} \lambda_j r_j \leq \sum_{i \in [M]} \mu_i h_i \right\}.$$

The MaxWeight computation in MWTA for single-category decomposable systems turns out to be the following integer linear program (ILP), which is related to knapsack problems.

$$\arg\max_{\Delta_{j,s}:j,s} \sum_{j,s} Q_{j,s} \Delta_{j,s}$$
s.t. 
$$\sum_{i} a_{j,s} r_{j,s} \le \sum_{i} u_{i} h_{i,s} \forall s \in [S],$$
(8)

This problem is an integer program, hence it is not clear whether this problem can be solved efficiently at all instants. In fact, it is a so-called unbounded knapsack problem for a given  $\mathbf u$  and  $\mathbf Q$ . This problem is known to be NP-hard [34]. There is a pseudo-polynomial algorithm based on dynamic programming which solves it exactly, but the runtime may depend on  $Q_{j,s}$ . This dynamic programming-based algorithm can be converted into a fully polynomial time approximation schemes (FPTAS) which achieves any  $(1-\epsilon)$  approximation of the problem in  $\operatorname{poly}\left(\frac{1}{\epsilon}\right)$  computations. Moreover, there exists faster greedy algorithms that achieve  $\frac{1}{2}$  approximation and can be converted into a polynomial time approximation scheme (PTAS) that achieves  $(1-\epsilon)$  approximation in  $\operatorname{poly}(n^{\frac{1}{\epsilon}})$  computations. Thus we can conclude that though the MWTA algorithm is computationally hard for single-category decomposable system, there exist efficient approximation schemes. It is not hard to show (Prop. 19 below) that an algorithm that gives  $(1-\epsilon)$  approximation of the optimization problem in MWTA can stabilize any  $\lambda$  for which  $\frac{\lambda}{(1-\epsilon)} \in \mathcal{C}_{\Gamma}^D$ .

For single-category non-decomposable systems, feasibility condition (2) of an allocation of  $a_j$  jobs of type j is given by,

$$\sum_{j} a_j r_{j,s} \le \sum_{i} u_i h_{i,s} \forall s \in [S], a_j \in \mathbb{Z}_+.$$

Hence the stabilizable region changes accordingly to

$$\mathcal{C}_{\Gamma} = \left\{ \sum_{\mathbf{u}} \Gamma(\mathbf{u}) \lambda(\mathbf{u}) : \lambda(\mathbf{u}) \in \mathcal{C}^{ND}(\mathbf{u}) \right\},$$

$$\mathcal{C}^{ND}(\mathbf{u}) = \operatorname{conv} \left\{ a_{j,s} \in \mathbb{Z}_{+} : \sum_{j} a_{j} r_{j,s} \leq \sum_{i} u_{i} h_{i,s} \text{ and } a_{j,s} = a_{j} \forall s \right\}.$$

where  $conv\{\cdot\}$  is the convex hull.

Hence, in this case, the MWTA allocation needs to solve

$$\arg \max_{\Delta_{j,s}:j} \sum_{j} \left( \sum_{s} Q_{j,s} \Delta_{j,s} \right)$$
s.t. 
$$\sum_{i} \Delta_{j,s} r_{j,s} \leq \sum_{i} u_{i} h_{i,s}, \Delta_{j,s} = \Delta_{j,s'}, \text{ for all } s, s' \in [S],$$
(9)

and then divide the allocated jobs arbitrarily among agents while meeting their per skill time-availability, as there is only one category of agents.

This problem is also a knapsack-like integer program, with an additional constraint that the number of (j, s)-items has to be the same as the number of (j, s')-items for all j, s and s'. Such a problem is called a multi-dimensional knapsack problem. This problem is also NP-hard. Moreover, provably there cannot exist a fully polynomial time approximation scheme for this problem [34].

For a multi-dimensional knapsack problem, there exists an approximation scheme that achieves an approximation factor equal to the dimension d [34]. This approximation scheme can be converted into a PTAS with complexity  $O(N^{\kappa})$  and an approximation factor of  $1 - \epsilon$ , where  $\kappa$  is strictly increasing with dimension and  $\frac{1}{\epsilon}$ . In the case of our setting, the number of skills S (dimension) is large and may scale with N, hence this scheme is not suitable.

Though the total number of skills S can scale with N, in most cases the number of skill-parts that a type j job has is a constant, i.e.,  $r_j$  has most coordinates as 0. Thus it is apparent from the objective function that the optimal choice of  $\Delta_{j,s}$  is 0 for the corresponding coordinates s. This allows us to rewrite the optimization problem as another multi-dimensional knapsack problem with constant dimensions given by  $\max_j |\{s: r_{j,s} > 0\}|$ .

For this problem we can use the PTAS to obtain arbitrarily close approximation and hence can stabilize a rate-region arbitrarily close to  $\mathcal{C}^{ND}$ . But the complexity of this algorithm is very high as complexity scales super-exponentially  $(N^k)$  with the approximation factor (unlike decomposable systems where we have an FPTAS, polynomial in  $\frac{1}{2}$ ).

Note that our goal is not to solve (9) optimally, but to have a fast allocation scheme that can stabilize a large fraction of  $\mathcal{C}^{ND}$ . In this regard we can take a different approach that exploits basic characteristics of a crowd system. Since N and  $\lambda_j(N)$  for  $j \in [N]$  are large in most crowd systems, if an allocation scheme stabilizes any rate  $\lambda$  for  $\lambda + c\mathbf{1} \in \mathcal{C}$ , then it stabilizes  $\left(1 - \max_i \frac{1}{\lambda_i(N)}\right)\mathcal{C}$ . Note that as  $\lambda_i(N)$  scales with N, this implies that such a scheme would stabilize almost all of  $\mathcal{C}$ . Motivated by this, we propose the following allocation scheme, which is a modification of (9).

$$\{\hat{x}_{j}, j\} = \arg \max_{x_{j} \in \mathbb{R}: j} \sum_{j} \left( \sum_{s} Q_{j,s} \right) x_{j}$$
s.t. 
$$\sum_{i} x_{j} r_{j,s} \leq \sum_{i} u_{i} h_{i,s}, \text{ for all } s,$$

$$(10)$$

and allocate  $\tilde{\Delta}_j = \lfloor \hat{x}_j \rfloor$  jobs of type j to the agents, splitting arbitrarily while meeting time-availability constraints.

Note that since in (10), the variables are relaxed to  $\mathbb{R}$  from  $\mathbb{Z}$ ,  $\sum_{j} \left(\sum_{s} Q_{j,s}\right) \hat{x}_{j} \geq \sum_{j} \left(\sum_{s} Q_{j,s}\right) \hat{z}_{j}$ . Again,  $\tilde{\Delta}_{j} = \lfloor \hat{x}_{j} \rfloor \geq \hat{x}_{j} - 1$ , hence  $\sum_{j} \left(\sum_{s} Q_{j,s}\right) \tilde{\Delta}_{j} \geq \sum_{j} \left(\sum_{s} Q_{j,s}\right) \left(\hat{z}_{j} - 1\right)$ .

The following proposition guarantees that a proposed LP-relaxation scheme stabilizes any  $\lambda$  with  $\lambda + 1 \in \mathcal{C}^{ND}$ .

**Proposition 19.** Let  $\mathcal{P}$  be an allocation scheme that at epoch t does an allocation  $\{\Delta(t)\}$  instead of  $\{\hat{\Delta}(t)\}$  of the MWTA allocation scheme, which satisfies

$$\sum_{j,s} Q_{j,s}(t)\hat{\Delta}_{j,s}(t) \le \sum_{j,s} Q_{j,s}(t)\Delta_{j,s}(t) + \sum_{j,s} Q_{j,s}(t)\delta,$$

or,

$$(1 - \epsilon) \sum_{j,s} Q_{j,s}(t) \hat{\Delta}_{j,s}(t) \le \sum_{j,s} Q_{j,s}(t) \Delta_{j,s}(t),$$

stabilizes any rate  $\lambda \in C$  if  $\lambda + \delta \mathbf{1} \in C$  or  $\frac{1}{1-\epsilon}\lambda \in C$  respectively.

*Proof:* This proof follows the same steps as the proof of Thm. 8. We first prove the result for an allocation with

$$\sum_{j,s} Q_{j,s}(t)\hat{\Delta}_{j,s}(t) \le \sum_{j,s} Q_{j,s}(t)\Delta_{j,s}(t) + \sum_{j,s} Q_{j,s}(t)\delta.$$

as follows:

$$\mathbb{E}\left[L\left(\mathbf{Q}(t+1)\right) - L\left(\mathbf{Q}(t)\right)|\mathbf{Q}(t)\right] \\ \leq \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)A_{j}(t)|\mathbf{Q}(t)\right] - \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)\left(\hat{\Delta}_{j,s}(t) - \delta\right)|\mathbf{Q}(t)\right] + \mathbb{E}\left[\sum_{j,s} \left(A_{j}^{2}(t) + (\hat{\Delta}_{j,s} - \delta)^{2}\right)|\mathbf{Q}(t)\right] \\ \leq \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)A_{j}(t)|\mathbf{Q}(t)\right] - \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)\left(\hat{\Delta}_{j,s}(t) - \delta\right)|\mathbf{Q}(t)\right] + \mathbb{E}\left[\sum_{j,s} \left(A_{j}^{2}(t) + (\hat{\Delta}_{j,s} - \delta)^{2}\right)|\mathbf{Q}(t)\right] \\ \leq \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)A_{j}(t)|\mathbf{Q}(t)\right] - \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)\left(\hat{\Delta}_{j,s}(t) - \delta\right)|\mathbf{Q}(t)\right] + \mathbb{E}\left[\sum_{j,s} \left(A_{j}^{2}(t) + (\hat{\Delta}_{j,s} - \delta)^{2}\right)|\mathbf{Q}(t)\right] \\ \leq \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)A_{j}(t)|\mathbf{Q}(t)\right] - \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)\left(\hat{\Delta}_{j,s}(t) - \delta\right)|\mathbf{Q}(t)\right] + \mathbb{E}\left[\sum_{j,s} \left(A_{j}^{2}(t) + (\hat{\Delta}_{j,s} - \delta)^{2}\right)|\mathbf{Q}(t)\right] \\ \leq \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)A_{j}(t)|\mathbf{Q}(t)\right] - \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)\left(\hat{\Delta}_{j,s}(t) - \delta\right)|\mathbf{Q}(t)\right] + \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t) + (\hat{\Delta}_{j,s} - \delta)^{2}\right] + \mathbb{E}\left[\sum_{j,$$

We can bound the last term as above, because  $(\hat{\Delta}_{j,s} - \delta)^2 \leq \hat{\Delta}_{j,s}^2 + \delta^2$ . Thus,

$$\mathbb{E}\left[L\left(\mathbf{Q}(t+1)\right) - L\left(\mathbf{Q}(t)\right)|\mathbf{Q}(t)\right] \leq B + \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)(A_{j}(t)+\delta)|\mathbf{Q}(t)\right] - \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)\hat{\Delta}_{j,s}(t)|\mathbf{Q}(t)\right]$$

Now note that if  $\lambda$  is such that  $\lambda + \delta \mathbf{1} \in \mathcal{C}$ , then we can write it in terms of convex combinations of  $\mathbf{d} \in C(\mathbf{u})$  and follow the same steps as in the proof of Thm. 8.

Similarly for the other case of constant factor approximation we have:

$$\mathbb{E}\left[L\left(\mathbf{Q}(t+1)\right) - L\left(\mathbf{Q}(t)\right)|\mathbf{Q}(t)\right] \leq B + \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)A_{j}(t)|\mathbf{Q}(t)\right] - \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)\hat{\Delta}_{j,s}(t)(1-\epsilon)|\mathbf{Q}(t)\right]$$

as  $(1 - \epsilon)^2 \hat{\Delta}_{i,s}^2 \leq \hat{\Delta}_{i,s}^2$ .

If  $\lambda \in (1-\epsilon)\mathcal{C}$ , then by definition of  $\mathcal{C}$  and  $(1-\epsilon)\mathcal{C}$ , there exist  $\nu(\mathbf{u}) \in \mathcal{C}(\mathbf{u})$  and  $\{\mathbf{d}_k(\mathbf{u}) \in C(\mathbf{u})\}$  such that,  $\lambda \leq (1-\epsilon)\nu(\mathbf{u})\Gamma(u)$  and  $\sum_k \gamma_k \mathbf{d}^k = \nu(\mathbf{u})$  for  $\gamma_k > 0$ ,  $\sum_k \gamma_k = 1$ . As for any  $\mathbf{d}^k$  for a given  $\mathbf{u}$ ,  $\sum_{j,s} Q_{j,s}(t)\hat{\Delta}_{j,s}(t) \leq \sum_{j,s} Q_{j,s}(t)\mathbf{d}_{j,s}^k$ ,  $\sum_{j,s} Q_{j,s}(t)\lambda_j \leq \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)\hat{\Delta}_{j,s}(t)|\mathbf{Q}(t)\right]$ . Then following the proof of Thm. 8, the result follows.

## APPENDIX II

In this appendix, we present proofs of the main results in Secs. III–V. As mentioned earlier, most of these results extend to systems with stationary and ergodic arrival and availability processes, but here we only present results for i.i.d. processes.

# A. Proof of Theorem 5

Here we only prove the converse part, i.e.,  $\lambda \notin \mathcal{C}$  cannot be stabilized by any policy. For the direct part, it is sufficient to prove there exists a scheme that stabilizes any  $\lambda \in \mathcal{C}$ , and so the proof of Thm. 8 below is sufficient.

First we prove that  $\mathcal{C}$  is a convex subset of  $\mathbb{R}_+^N$ . If  $\boldsymbol{\lambda}, \boldsymbol{\lambda}' \in \mathcal{C}$ , then there exist  $(\boldsymbol{\lambda}(\mathbf{u}) \in \mathcal{C}(\mathbf{u}) : \mathbf{u} \in \mathbb{Z}_+^M)$  and  $(\boldsymbol{\lambda}'(\mathbf{u}) \in \mathcal{C}(\mathbf{u}) : \mathbf{u} \in \mathbb{Z}_+^M)$  such that

$$\sum_{\mathbf{u}}\Gamma(\mathbf{u})\boldsymbol{\lambda}(\mathbf{u})=\boldsymbol{\lambda},\quad \sum_{\mathbf{u}}\Gamma(\mathbf{u})\boldsymbol{\lambda}'(\mathbf{u})=\boldsymbol{\lambda}'.$$

Thus for any  $\gamma \in [0, 1]$ ,

$$\gamma \boldsymbol{\lambda} + (1 - \gamma) \boldsymbol{\lambda}' = \sum_{\mathbf{u}} \Gamma(\mathbf{u}) (\gamma \boldsymbol{\lambda}(\mathbf{u}) + (1 - \gamma) \boldsymbol{\lambda}'(\mathbf{u}).$$

Note that  $C(\mathbf{u})$  is convex since it is the convex hull of  $C(\mathbf{u})$ ; hence  $\gamma \lambda(\mathbf{u}) + (1 - \gamma)\lambda'(\mathbf{u}) \in C(\mathbf{u})$ , which in turn implies  $\gamma \lambda + (1 - \gamma)\lambda' \in C$ . This proves convexity of C.

Thus  $\bar{\mathcal{C}}$  is a closed convex set. Hence for any  $\lambda^O \notin \bar{\mathcal{C}}$ , there exists a hyperplane  $h^T x = c$  that separates  $\bar{\mathcal{C}}$  and  $\lambda^O$ , i.e., there exists an  $\epsilon > 0$  for any  $\lambda \in \bar{\mathcal{C}}$  such that  $h^T \lambda^O \geq h^T \lambda + \epsilon$ .

Hence under any policy:

$$\mathbb{E}\left[h^{T}\mathbf{Q}(t+1)\right] = \mathbb{E}\left[h^{T}\left(\mathbf{Q}(t) + \mathbf{A}(t) - \mathbf{D}(t)\right)\right]$$
$$= \mathbb{E}\left[h^{T}\left|\mathbf{Q}(t) + \mathbf{A}(t) - \mathbf{\Delta}(t)\right|^{+}\right]$$

where  $\Delta(t)$  is the number of possible departure under the scheme if there were infinite number of jobs of each type, and  $|\cdot|^+$  is shorthand for  $\max(\cdot,0)$ . As  $|x|^+$  is a convex function of x,  $h^T |\mathbf{Q}(t) + \mathbf{A}(t) - \mathbf{\Delta}(t)|^+$  is a convex function of  $\mathbf{Q}(t)$ ,  $\mathbf{A}(t)$ , and  $\mathbf{\Delta}(t)$ . Thus by Jensen's inequality:

$$\mathbb{E}\left[h^{T}\left|\mathbf{Q}(t)+\mathbf{A}(t)-\boldsymbol{\Delta}(t)\right|^{+}\right] \geq h^{T}\left|\mathbb{E}\left[\mathbf{Q}(t)\right]+\mathbb{E}\left[\mathbf{A}(t)\right]-\mathbb{E}\left[\boldsymbol{\Delta}(t)\right]\right|^{+}$$

Note that any  $\lambda$  is a  $\Gamma(\mathbf{u})$ -combination of some  $\{\lambda(\mathbf{u}) \in \mathcal{C}(\mathbf{u})\}$  and any  $\lambda(\mathbf{u})$  is some convex combination of elements of  $C(\mathbf{u})$ . Also, from the allocation constraints it is apparent that if  $\mathbf{a} \in C(\mathbf{u})$  then also  $\mathbf{a}' \in C(\mathbf{u})$  if  $\mathbf{a}' \leq \mathbf{a}$ . These two imply that for any  $\lambda \in \bar{\mathcal{C}}$ , if there exists a  $\lambda' \leq \lambda$  (component-wise) and  $\lambda' \geq \mathbf{0}$ , then  $\lambda' \in \bar{\mathcal{C}}$ . That is  $\mathcal{C}$  is *coordinate convex*. This in turn implies that for any  $\lambda^O \notin \bar{\mathcal{C}}$  there exists an  $h \neq \mathbf{0} \in \mathbb{R}_+^N$  such that hyperplane separation holds for this h. Thus for  $h \geq \mathbf{0}$ :

$$\mathbb{E}\left[h^{T}\mathbf{Q}(t+1)\right] \geq h^{T} \left|\mathbb{E}\left[\mathbf{Q}(t)\right] + \mathbb{E}\left[\mathbf{A}(t)\right] - \mathbb{E}\left[\mathbf{\Delta}(t)\right]\right|^{+}$$

$$= \sum_{j} \left|h_{j}\mathbb{E}\left[Q_{j}(t)\right] + h_{j}\mathbb{E}\left[A_{j}(t)\right] - h_{j}\mathbb{E}\left[\Delta_{j}(t)\right]\right|^{+}$$

$$\geq \sum_{j} \left(h_{j}\mathbb{E}\left[Q_{j}(t)\right] + h_{j}\mathbb{E}\left[A_{j}(t)\right] - h_{j}\mathbb{E}\left[\Delta_{j}(t)\right]\right)$$

$$\geq \mathbb{E}\left[h^{T}\mathbf{Q}(t)\right] + h^{T}\mathbb{E}\left[\boldsymbol{\lambda}^{O}\right] - \sup_{\boldsymbol{\lambda} \in \bar{\mathcal{C}}} h^{T}\boldsymbol{\lambda}$$

$$\geq \mathbb{E}\left[h^{T}\mathbf{Q}(t)\right] + \epsilon$$

Thus we have  $\mathbb{E}\left[h^T\mathbf{Q}(t+1)\right]\to\infty$ . As  $h\geq 0$ , this implies there exists j such that  $\mathbb{E}[Q_j(t)]\to\infty$  as  $t\to\infty$ . Hence, the system is not stable.

## B. Proof of Theorem 6

Consider the dynamics of  $Q_{j,s}(t)$ , the unallocated (j,s) tasks at the end of epoch t.

$$Q_{j,s}(t+1) = |Q_{j,s}(t) + A_j(t) - D_{j,s}(t)|^+$$

$$\geq Q_{j,s}(t) + A_j(t) - D_{j,s}(t)$$

$$\geq \sum_{k=0}^{t} (A_j(t) - D_{j,s}(t)).$$

As  $r_{j,s} \geq 0$ ,

$$\mathbb{E}[r_{j,s}Q_{j,s}(t+1)] \ge \sum_{k=0}^{t} \mathbb{E}[A_{j}(t)r_{j,s} - r_{j,s}D_{j,s}(t)]$$
$$= \sum_{k=0}^{t} (\lambda_{j}r_{j,s} - r_{j,s}\mathbb{E}[D_{j,s}(t)])$$

Consider any set  $J \subset [N]$ , then at any epoch t, to schedule a certain number of tasks of type (j, s), the system needs that much available usable skill-hours. This follows from conditions (1) and (2) and can be written as:

$$\sum_{j \in J} r_{j,s} D_{j,s}(t) \le \sum_{l \in \mathcal{N}(J)} \sum_{i \in [M^l]} h_{i,s}^l U_{i,s}^l.$$

This in turn implies

$$\sum_{j \in J} r_{j,s} \mathbb{E}\left[D_{j,s}(t)\right] \le \sum_{l \in \mathcal{N}(J)} \sum_{i \in [M^l]} h_{i,s}^l \mu_{i,s}^l.$$

Hence,

$$\mathbb{E}\left[\sum_{j\in J} r_{j,s} Q_{j,s}(t+1)\right] \ge \sum_{k=0}^{t} \left(\sum_{j\in J} \lambda_{j} r_{j,s} - \sum_{j\in J} r_{j,s} \mathbb{E}\left[D_{j,s}(t)\right]\right)$$

$$\ge \sum_{k=0}^{t} \left(\sum_{j\in J} \lambda_{j} r_{j,s} - \sum_{l\in \mathcal{N}(J)} \sum_{i\in [M^{l}]} h_{i,s}^{l} \mu_{i,s}^{l}\right)$$

$$(11)$$

For any  $\lambda \notin \bar{\mathcal{C}}^{out}$ , by definition there exists a  $J \subset [N]$  such that  $\sum_{j \in J} \lambda_j r_{j,s} - \sum_{l \in \mathcal{N}(J)} \sum_{i \in [M^l]} h_{i,s}^l \mu_{i,s}^l > 0$ . Thus in that case,  $\limsup_{t \to \infty} \mathbb{E}\left[\sum_{j \in J} r_{j,s} Q_{j,s}(t+1)\right] = \infty$ . Note that since J is finite and so is  $\max_j r_{j,s}$ , there exists a  $j \in J$  such that  $\limsup_{t \to \infty} \mathbb{E}\left[Q_{j,s}(t+1)\right] = \infty$ . This shows the system is not stable for  $\lambda \notin \bar{\mathcal{C}}^{out}$  and proves  $\mathcal{C}_{\Gamma} \subset \mathcal{C}^{out}$ .

# C. Proof of Proposition 7

We consider FD, FND, and IND cases separately.

- FD: The MaxWeight part chooses  $z_{j,s}^l$  to be integral which implies that integral number of tasks can be allocated if done appropriately. As hours are allocated from tasks in order, a task later in the order only gets allocated (partially or fully) after the tasks before it are fully allocated. This leads to no partially-allocated tasks.
- FND: Same ordering is used for all (j,s) tasks and MaxWeight chooses  $z_{j,s}^l$  such that  $a_{j,s}=a_{j,s'}$  (as it satisfies FND) and hence if an s-task of a job is chosen then also s' is chosen for  $r_{j,s}, r_{j,s'} > 0$ .
- IND: Same ordering is used for all (j,s) tasks, allocations to different categories are in same order (l=1 to L) and MaxWeight chooses  $z_{j,s}^l$  such that  $z_{j,s}^l = z_{j,s'}^l$  for all l,l' (as it satisfies IND), hence if a task of a job is allocated to category l then so are the other tasks.

# D. Proof of Theorem 8

Note that MaxWeight chooses an allocation  $\{\hat{\Delta}_{j,s}(t)\}$ . But, the maximum number of (j,s)-tasks that can be served is  $Q_{j,s}(t)+A_j(t)$ . By Prop. 7, Task Allocation does a feasible allocation for FD, FND, and IND systems. Also, note that in the Task Allocation algorithm, the number of allocated (j,s)-tasks is  $\hat{D}_{j,s}=\min\left(\hat{\Delta}_{j,s}(t),Q_{j,s}(t)+A_j(t)\right)$ .

Consider the usual Lyapunov function  $L(\mathbf{Q}) = \sum_{j,s} Q_{j,s}^2$ .

$$\begin{split} & \mathbb{E}\left[L\left(\mathbf{Q}(t+1)\right) - L\left(\mathbf{Q}(t)\right)|\mathbf{Q}(t)\right] \\ & = \mathbb{E}\left[\sum_{j,s}\left(Q_{j,s}^{2}(t+1) - Q_{j,s}^{2}(t)\right)|\mathbf{Q}(t)\right] \\ & = \mathbb{E}\left[\sum_{j,s}\left(\left(Q_{j,s}(t) + A_{j}(t) - \hat{D}_{j,s}\right)^{2} - Q_{j,s}^{2}(t)\right)|\mathbf{Q}(t)\right] \\ & \leq \mathbb{E}\left[\sum_{j,s}\left(\left(Q_{j,s}(t) + A_{j}(t) - \hat{\Delta}_{j,s}\right)^{2} - Q_{j,s}^{2}(t)\right)|\mathbf{Q}(t)\right] \\ & \leq \mathbb{E}\left[\sum_{j,s}\left(\left(Q_{j,s}(t) + A_{j}(t) - \hat{\Delta}_{j,s}\right)^{2} - Q_{j,s}^{2}(t)\right)|\mathbf{Q}(t)\right] \\ & \leq \mathbb{E}\left[\sum_{j,s}Q_{j,s}(t)A_{j}(t)|\mathbf{Q}(t)\right] - \mathbb{E}\left[\sum_{j,s}Q_{j,s}(t)\hat{\Delta}_{j,s}(t)|\mathbf{Q}(t)\right] + \mathbb{E}\left[\sum_{j,s}\left(A_{j}^{2}(t) + \hat{\Delta}_{j,s}^{2}\right)|\mathbf{Q}(t)\right] \end{split}$$

We bound the last term first.

$$\mathbb{E}\left[\sum_{j,s} \left(A_{j}^{2}(t) + \hat{\Delta}_{j,s}^{2}\right) | \mathbf{Q}(t)\right] = \sum_{j,s} \mathbb{E}[A_{j}^{2}] + \mathbb{E}\left[\sum_{j,s} \left(r_{j,s}\hat{\Delta}_{j,s}\right)^{2}\right] \\
\leq \sum_{j,s} \mathbb{E}[A_{j}^{2}] + \frac{1}{\min(r_{j,s} > 0)} \mathbb{E}\left[\left(\sum_{j,s} r_{j,s}\hat{\Delta}_{j,s}\right)^{2}\right] \\
\leq \sum_{j,s} \mathbb{E}[A_{j}^{2}] + \frac{1}{\min(r_{j,s} > 0)} \mathbb{E}\left[\left(\sum_{i,l} \left(\sum_{s} h_{i,s}^{l}\right) U_{i}^{l}\right)^{2}\right] \\
\leq \sum_{j,s} \mathbb{E}[A_{j}^{2}] + \frac{\left(\max_{l,i,s} h_{i,s}^{l}\right)^{2}}{\min(r_{j,s} > 0)} \max_{l,i} \mathbb{E}\left[\left(U_{l}^{l}\right)^{2}\right] \frac{M(M+1)}{2} \tag{13}$$

This is a constant  $B < \infty$  independent of  $\mathbf{Q}$ , as  $\mathbb{E}[A_j^2]$  and  $\mathbb{E}\left[\left(U_i^l\right)^2\right]$  are finite for all j, l, i.

To bound the first term, note that if  $\lambda + \epsilon \mathbf{1} \in \mathcal{C}$ , then there exist  $\{\nu(\mathbf{u}) \in \mathcal{C}(\mathbf{u})\}$  such that  $\lambda_j \leq \sum_{\mathbf{u}} \Gamma(\mathbf{u})\nu(\mathbf{u}) - \epsilon$  for all  $j \in [J]$ . Again note that as  $\mathcal{C}(\mathbf{u})$  is the convex hull of  $C(\mathbf{u})$ ,  $\nu(\mathbf{u}) = \sum_k \gamma_k \mathbf{d}^k(\mathbf{u})$  for some  $\{\mathbf{d}^k(\mathbf{u}) \in C(\mathbf{u})\}$  and  $\gamma_k \geq 0$  with  $\sum_k \gamma_k = 1$ . So,

$$\mathbb{E}\left[\sum_{j,s} Q_{j,s}(t)A_{j}(t)|\mathbf{Q}(t)\right] = \sum_{j,s} Q_{j,s}(t)\lambda_{j}$$

$$\leq \sum_{j,s} Q_{j,s}(t)\nu_{j} - \epsilon \sum_{j,s} Q_{j,s}$$

$$\leq \sum_{j,s} Q_{j,s}(t) \sum_{\mathbf{u}} \Gamma(\mathbf{u}) \sum_{k} \gamma_{k} d_{j}^{k}(\mathbf{u}) - \epsilon \sum_{j,s} Q_{j,s}$$

$$= \sum_{\mathbf{u}} \Gamma(\mathbf{u}) \sum_{j,s} Q_{j,s}(t) \sum_{k} \gamma_{k} d_{j}^{k}(\mathbf{u}) - \epsilon \sum_{j,s} Q_{j,s}$$

$$\leq \sum_{\mathbf{u}} \Gamma(\mathbf{u}) \max_{\mathbf{d}(\mathbf{u}) \in C(\mathbf{u})} \sum_{j,s} Q_{j,s}(t) d_{j,s}(\mathbf{u}) - \epsilon \sum_{j,s} Q_{j,s}$$

$$= \mathbb{E}\left[\sum_{j,s} Q_{j,s}(t) \hat{\Delta}_{j,s}(t)|\mathbf{Q}(t)\right] - \epsilon \sum_{j,s} Q_{j,s}.$$

Thus, we have a bound on the Lyapunov drift,

$$\mathbb{E}\left[L\left(\mathbf{Q}(t+1)\right) - L\left(\mathbf{Q}(t)\right)|\mathbf{Q}(t)\right] \le B - \epsilon \sum_{j,s} Q_{j,s}.$$

Hence,

$$\mathbb{E}\left[L\left(\mathbf{Q}(T)\right) - L\left(\mathbf{Q}(0)\right)\right] \le BT - \epsilon \sum_{t=0}^{T-1} \sum_{j,s} w_{j,s} \mathbb{E}[Q_{j,s}(t)].$$

As  $L(\mathbf{Q}(0)) < \infty$  and  $L(\mathbf{Q}) \ge 0$  for all  $\mathbf{Q}$ , we have that for all T,

$$\frac{1}{T} \sum_{t=0}^{T-1} \sum_{j,s} \mathbb{E}[Q_{j,s}(t)] \le \frac{B}{\epsilon} + \frac{L(0)}{T} < \infty$$

This in turn implies  $\limsup_{t\to\infty}\sum_{j,s}\mathbb{E}[Q_{j,s}(t)]<\infty$ , otherwise the time-average cannot be finite. This implies that for all (j,s) with  $r_{j,s}>0$ ,  $\limsup_{t\to\infty}\mathbb{E}[Q_{j,s}(t)]<\infty$ .

Again note that  $Q_j(t) \leq \sum_s Q_{j,s}$ , as there can be unallocated jobs with more than one part unallocated. Hence,  $\limsup_{t\to\infty} \mathbb{E}[Q_j(t)] < \infty$  for all  $j\in[N]$ .

# E. Proof of Theorem 11

In the GreedyAgent algorithm, as each agent with available skill-hours greedily chooses to serve a task, no (j, s) task of size r can remain unallocated if there is an agent (or agents) with s skill-hour (total) of at least r. Since at each allocation epoch a task should either be allocated totally or not at all (i.e., x < r hours cannot be allocated), it may happen that some agent hours are wasted, as that does not meet the task allocation requirement.

Note that since any job requirement is less than  $\bar{r} = \max_{j,s} r_{j,s}$ , at most  $\bar{r}$  agent-skill-hours can be wasted. Let  $H_s(t)$  be the process of unallocated job-hours for skill s after the allocation at epoch t. Then for all t,

$$H_s(t+1) \le \left| H_s(t) + \sum_j A_j(t) r_{j,s} - \sum_i U_i(t) h_{i,s} + \bar{r} \right|^+$$
.

This implies that process  $G_s(t)$  given by  $G_s(t+1) = \left|G_s(t) + \sum_j A_j(t) r_{j,s} - \sum_i U_i(t) h_{i,s} + \bar{r}\right|^+$  bounds  $H_s(t)$ .  $G_s(t)$  has dynamics of a queue with arrival process  $X_s(t) = \sum_j A_j(t) r_{j,s} + \bar{r}$  and departure process  $Y_s(t) = \sum_i U_i(t) h_{i,s}$ . Let  $A_j(\theta) = \mathbb{E}\left[e^{\theta A_j(t)}\right]$  and  $U_i(\theta) = \mathbb{E}\left[e^{\theta U_i(t)}\right]$  for  $j \in [N]$  and  $i \in [M]$ . For  $\theta \in \mathbb{R}$ , then,

$$\mathbb{E}\left[e^{\theta(X_{s}(t)-Y_{s}(t))}\right] = \mathbb{E}\left[e^{\theta X_{s}(t)}\right] \mathbb{E}\left[e^{-\theta Y_{s}(t)}\right] \\
= \mathbb{E}\left[e^{\theta \sum_{j} A_{j}(t)r_{j,s}+\bar{r}}\right] \mathbb{E}\left[e^{-\theta \sum_{i} U_{i}(t)h_{i,s}}\right] \\
= e^{\theta \bar{r}} \prod_{j} \mathbb{E}\left[e^{\theta A_{j}(t)r_{j,s}}\right] \prod_{i} \mathbb{E}\left[U_{i}(t)h_{i,s}\right] \\
= e^{\theta \bar{r}} \prod_{j} \mathcal{A}_{j}(\theta r_{j,s}) \prod_{i} \mathcal{U}_{i}(-\theta h_{i,s}) \\
= \exp\left(\theta \bar{r} + \sum_{j} \log \mathcal{A}_{j}(\theta r_{j,s}) + \sum_{i} \log \mathcal{U}_{i}(-\theta h_{i,s})\right).$$
(14)

First consider the Gaussian-dominated case. Since the process variance is no more than the mean and the moment generating function of the variance is upper-bounded by that of a zero-mean Gaussian:

$$\log \mathcal{A}_{j}(\theta r_{j,s}) \leq \lambda_{j} \theta r_{j,s} + \lambda_{j} \frac{(\theta r_{j,s})^{2}}{2}$$
$$\log \mathcal{U}_{i}(-\theta h_{i,s}) \leq -\mu_{j} \theta h_{i,s} + \mu_{j} \frac{(\theta h_{i,s})^{2}}{2}.$$

Note that for any two functions  $k_1x^2$  and  $k_2x$ ,  $\lim_{x\to 0}k_2x/k_1x^2=\infty$ , and hence for any  $\epsilon\in(0,1)$  there exists  $x^*>0$  such that for all  $x< x^*$ ,  $k_1x^2/k_2x<\epsilon$ . Hence for any  $\epsilon\in(0,1)$ , there exist  $\theta_{j,s}^*,\theta_{i,s}^*>0$ , for all i,j,s such that for all  $\theta<\theta^*=\min_{i,j,s}(\theta_{i,s}^*,\theta_{i,s}^*)$ ,

$$\log \mathcal{A}_{i}(\theta r_{i,s}) \le \lambda_{i} \theta^{*} r_{i,s} (1 + \epsilon) \tag{15}$$

$$\log \mathcal{U}_i(-\theta h_{i,s}) \le -\mu_i \theta h_{i,s}(1-\epsilon) \tag{16}$$

Note that since N, S, and M are finite and  $\theta_{j,s}^*, \theta_{i,s}^* > 0$ , for all  $i, j, s, \theta^* > 0$ . Moreover, note that  $\theta^*$  does not depend on  $\lambda, \mu$  since the ratio of the linear and quadratic terms in the log moment generating functions are independent of  $\lambda$  and  $\mu$ .

As  $e^{\theta} - 1 = \sum_{k=1}^{\infty} \frac{\hat{\theta}^k}{k!}$ , for the Poisson-dominated case we have

$$\log \mathcal{A}_{j}(\theta r_{j,s}) \leq \lambda_{j} \sum_{k} \frac{(\theta r_{j,s})^{k}}{k!}$$
$$\log \mathcal{U}_{i}(-\theta h_{i,s}) \leq \mu_{j} \sum_{k} \frac{(-\theta h_{i,s})^{k}}{k!}$$

Again, by the same argument, we can have a  $\theta^*$  for which (26) and (27) are satisfied. Thus, for all  $\theta < \theta^*$  we have:

$$\mathbb{E}\left[e^{\theta(X_{s}(t)-Y_{s}(t))}\right] \leq \exp\left(\theta\bar{r} + \sum_{j} \lambda_{j}\theta^{*}r_{j,s}(1+\epsilon) - \sum_{i} \mu_{i}\theta h_{i,s}(1-\epsilon)\right)$$

$$\leq \exp\left(\theta\left(\bar{r} - \sum_{i} \mu_{i}h_{i,s}(\alpha-\epsilon)\right)\right). \tag{17}$$

Note (28) follows from the fact  $\lambda \in (1-\alpha)\mathcal{C}^{out}$ . As  $\epsilon>0$  can be chosen arbitrarily small, we can have  $\alpha-\epsilon>0$ . Since  $\sum_i \mu_i h_{i,s} > \sum_j (1-\alpha)\lambda_j r_{j,s}$  and  $\sum_j \lambda_j r_{j,s}$  scales with  $\lambda(N)$ , for sufficiently large  $\lambda_\alpha$  with  $\lambda_j \geq \lambda_\alpha$  for all j, we have  $\bar{r} - \sum_i \mu_i h_{i,s} (\alpha - \epsilon) \leq -\gamma \sum_i \mu_i h_{i,s} (\alpha - \epsilon)$ , for some  $\gamma>0$ . Thus, we have for some  $\theta>0$ ,

$$\mathbb{E}\left[e^{\theta(X_s(t)-Y_s(t))}\right] \le \exp\left(-\theta K\right),\tag{18}$$

where K scales with  $\lambda$ .

Now by Loynes' construction:

$$G_s(t) = \max_{\tau < t} \sum_{k=\tau}^{t} (X_s(t) - Y_s(t)).$$

Hence,

$$\Pr(G_{s} > g) \leq \frac{\mathbb{E}\left[e^{\theta G_{s}}\right]}{e^{\theta g}}$$

$$\leq e^{-\theta g} \mathbb{E}\left[e^{\theta \max_{\tau} \sum_{t=0}^{\tau} (X_{s}(t) - Y_{s}(t))}\right]$$

$$\leq e^{-\theta g} \sum_{\tau=1}^{\infty} \left(\mathbb{E}\left[e^{\theta(X_{s}(t) - Y_{s}(t))}\right]\right)^{\tau}$$

$$\leq e^{-\theta g} \sum_{\tau=1}^{\infty} \exp\left(-\theta \tau K\right)$$

$$\leq e^{-\theta g} \frac{1}{1 - \exp\left(-\theta K\right)}.$$
(19)

This in turn implies

$$\Pr\left(\max_{s} G_{s} > c \log N\right) \leq S \Pr\left(G_{s} > c \log N\right)$$

$$\leq \frac{Se^{-\theta c \log N}}{1 - \exp\left(-\theta K\right)}$$

$$\leq \frac{S}{N^{\theta c}} \frac{1}{1 - \exp\left(-\theta K\right)}.$$

Note that the total number of unallocated jobs in the system is upper-bounded by  $(\min_{j,s:r_{j,s}>0} r_{j,s})^{-1} \sum_s G_s$ , as any unallocated job has at least  $\min_{j,s:r_{j,s}>0} r_{j,s}$  skill-hours unallocated. Hence the total number of unallocated tasks in the system, Q, satisfies:

$$\Pr\left(Q > cS \log N\right) \le \Pr\left(\sum_{s} G_{s} > c \min_{j,s:r_{j,s} > 0} r_{j,s} S \log N\right)$$

$$\le \Pr\left(\max_{s} G_{s} > c \min_{j,s:r_{j,s} > 0} r_{j,s} \log N\right) \frac{S}{N^{\theta c \min_{j,s:r_{j,s} > 0} r_{j,s}}} \frac{1}{1 - \exp\left(-\theta K\right)}.$$

Note that  $\theta^* > 0$  does not depend on  $\lambda$ , so  $\frac{1}{1 - \exp(-\theta K)}$  is O(1). As S = O(N), we can choose a c > 0 such that  $\theta^* c \min_{j,s:r_{j,s}>0} r_{j,s} > 3$  and hence,  $\Pr\left(Q > cS \log N\right) \leq o(N^{-2})$ .

# F. Proof of Proposition 12

We prove this proposition by constructing a simple (but general) system and show that the system is not stable via a domination argument with a Markov chain that is not positive recurrent.

Consider  $N=1,\ M=2,\ \text{and}\ S>1$  being even. Let  $r=1,\ h_1=(1,1,\cdots,\frac{S}{2}\ \text{terms},0,0,\cdots)$  and  $h_2=(0,0,\cdots,\frac{S}{2}\ \text{terms},1,1,\cdots)$ . Let the arrival to the system be i.i.d. with

$$\begin{cases} (1 - \alpha - \delta)\lambda, & \text{w.p. } (1 - \epsilon) \\ 2\lambda, & \text{w.p. } \epsilon, \end{cases}$$

such that  $(1+\alpha)\epsilon - \delta(1-\epsilon) = 0$ , resulting in mean arrival rate  $(1-\lambda)$  and variance  $<\lambda$ . One possible construction is to take  $\epsilon = \frac{1}{2\lambda}$  and  $\delta$  accordingly. The agent-availability process is considered to be  $U_1 = U^2 = \lambda$ . Note that both arrival and agent availability processes are Gaussian-dominated as well as Poisson-dominated.

First, consider the case where greedy picking of tasks by the agents may be adversarial. Each agent picks all the tasks that it can take from the job and so agents of different types pick from a different half of the S skill-parts. Thus if there are  $\geq 2\lambda$  jobs, in worst case (where adversary gives the job to the agent) agents of type 1 may pick  $\frac{S}{2}$  job-parts of  $\lambda$  jobs, while agents of type 2 pick parts of other  $\lambda$  jobs. Hence no job is actually allocated at that allocation epoch. By the next epoch at least  $(1-\alpha)\lambda$  jobs have come and if the agents pick jobs in an adversarial manner, again no job is actually allocated. Thus the number of unallocated jobs keep growing after it hits  $2\lambda$  once. Note that since there is a positive probability of  $\geq 2\lambda$  arrivals, with strictly positive probability, the number of unallocated jobs grows without bound.

Next, we prove the case where greedy picking of the tasks by the agents is random, i.e., an agent picks all S/2 parts of a randomly selected job (without replacement). As arrivals and availability are i.i.d., we can describe the number of unallocated jobs by a Markov chain Q(t). Note that for  $Q \le \lambda$ ,  $\Pr(Q \to 0) = 1$ . On the other hand,  $0 < \Pr(0 \to Q) < 1$  for  $Q < 2\lambda$ .

Consider any  $Q = n\lambda$  for n > 1. Note that  $\Pr(Q \to x) = \text{for } x < (n-1)\lambda$  as no more than  $\lambda$  jobs can be scheduled, because there are  $\lambda$  agents of each type. Again,

$$\Pr\left(Q \text{ decreases at least by } 1\right) \leq 1 - \left(1 - \frac{\lambda}{Q + \lambda - \alpha\lambda}\right)^{\lambda}$$

because there are at least  $(1 - \alpha)\lambda$  arrivals and each type picks  $\lambda$  agents randomly and this is the probability that the picked sets have a non-empty intersection. Again, as there are  $2\lambda$  arrivals w.p.  $\geq \epsilon$  we have

$$\Pr\left(Q \text{ increases by at least } \lambda\right) \geq \epsilon \left(1 - \frac{\lambda}{Q + 2\lambda}\right)^{\lambda}.$$

Based on computation of transition probabilities for each transition, it follows that for all  $k \ge 1$ , the probability of Q decreasing by at least 1 as well as the probability of Q decreasing by k is decreasing with Q, whereas the probability of Q increasing by k increases.

Hence, we can dominate the above chain by another chain  $\hat{Q}$  on  $\lambda \mathbb{Z}^+$  with transition probabilities

$$\Pr(\lambda n \to \lambda(n+1)) = \epsilon \left(1 - \frac{1}{n+1}\right)^{\lambda}$$

$$\Pr(\lambda(n+1) \to \lambda n) = 1 - \left(1 - \frac{1}{n+1}\right)^{\lambda}.$$

Now the chain  $\hat{Q}(t)$  is a birth-death chain. If it has a finite (summable over states) invariant, then that is unique. We first assume that the invariant is  $\pi$  and then show that it is not summable to prove that it is not positive recurrent.<sup>4</sup> As this is a birth-death chain the invariant measure must satisfy:

$$\pi(n+1) = \pi(n) \frac{\epsilon \left(1 - \frac{1}{n+1}\right)^{\lambda}}{1 - \left(1 - \frac{1}{n+1}\right)^{\lambda}}.$$

Since

$$\frac{\epsilon \left(1 - \frac{1}{n+1}\right)^{\lambda}}{1 - \left(1 - \frac{1}{n+1}\right)^{\lambda}} \to \infty$$

as  $n \to \infty$  for any finite  $\lambda > 0$ , this shows that  $\pi$  is not finite.

# G. Proof of Theorem 13

Consider the different types of unallocated jobs. These are given by  $\{Q_i(t): j \in [N]\}$ .

Consider the following processes: for each  $s \in [S]$ ,  $Q^s(t) = \sum_{j:r_{j,s}>0} Q_j r_{j,s}$  which represent the number of unserved hours of skills s over all jobs.

We now construct another process  $\tilde{Q}$  s.t. it dominates the process  $\sum_s Q^s$ . So, if we can show upper-bound on  $\tilde{Q}$ , then the same bound applies for  $\sum_s Q^s$ . Hence, in turn we get a bound for  $\{Q_j(t)\}$  (as  $\min\{r_{j,s}>0\}=\Theta(1)$  by the assumption that  $\{r_{j,s}\}$  do not scale with the system size).

Towards constructing a suitable  $\tilde{Q}$  we make the following observation about the dynamics of  $Q^s$  and  $\{Q_j\}$ . At each time t,  $\sum_j A_{j,1}(t)r_{j,s}$  amount of s skill hour is brought to add to  $Q^s$ . Also, this queue gets some service depending on the available agent hours.

At time t,  $\sum_{m} U_m(t)h_{m,s}$  s-skill hour of service is brought by the agents.

For a job to be allocated, all tasks of it must find an allocation. Hence, for a job in type j-job to find an allocation it must get  $r_{j,s}$  hours of service from each skill s. Thus at any time t any skill s queue gets a service of at least

$$\min_{s \in [S]} \sum_{m} U_m h_{m,s} - \bar{r},$$

where  $\bar{r} = \max\{r_{j,k,s}\}$ . This is because of the following. For each skill  $\sum_s U_m h_{m,s}$  hour is available. Note that a job can be allocated if all its tasks find allocations, coverse of which is also true. That is if all tasks of a step found allocation then the step can be allocated. As  $\min_{s \in [S]} \sum_s U_m h_{m,s}$  hours of service is brought by the agents for each skill, at least  $\min_{s \in [S]} \sum_s U_m h_{m,s} - \bar{r}$  of s-skill hours are served (because a maximum of  $\bar{r}$  can be wasted, as no task is of size more than  $\bar{r}$ ).

Also, note that the amount of required service brought to the queue  $Q^s$  at time t is upper-bounded by

$$\max_{s \in [S]} \sum_{j} A_j(t) r_{j,s}$$

<sup>&</sup>lt;sup>4</sup>An alternate proof follows from noting that if we take a Lyapunov  $\hat{Q}$  itself, then it has bounded jumps and it is easy to check that after certain Q > 0 the drift is strictly positive, and invoke the Foster-Lyapunov (converse) theorem for irreducible chain with bounded absolute drift.

Consider a process  $\tilde{Q}^s$  with evolution

$$\tilde{Q}^{s}(t+1) = \max(\tilde{Q}^{s}(t) + \max_{s \in [S]} \sum_{j} A_{j}(t)r_{j,s} - \min_{s \in [S]} \sum_{m} U_{m}h_{m,s} + \bar{r}, 0).$$

Note that given  $\tilde{Q}^s(t_0) \ge Q^s(t_0)$  at some  $t_0$ , the same holds true for all  $t \ge t_0$ . This is because for  $x, a, b \ge 0$  and  $x', a', b' \ge 0$ , with  $x \ge x'$ ,  $a \ge a'$  and  $b \le b'$ 

$$\max(x + a - b, 0) \ge \max(x' + a' - b', 0)$$

and hence, the monotonicity propagates over time.

Thus, to bound  $\sum_s Q^s$  it is sufficient to bound  $\sum_s \tilde{Q}^s(t)$ . Note that each of  $\tilde{Q}^s$  has exactly same evolution, so let us consider

$$\tilde{Q} := S\tilde{Q}^1,$$

which bounds  $\sum_{s} Q^{s}$ .

From the evolution:

$$\tilde{Q}(t+1) = \max(\tilde{Q}(t) + S \max_{s \in [S]} \sum_{j} A_j(t) r_{j,s} - S \min_{s \in [S]} \sum_{m} U_m h_{m,s} + \bar{r}, 0)$$

we can write the Loynes' construction for this process which has the same distribution as this process (and for simplicity we use the same notation, as we are interested in the distribution).

$$\tilde{Q}^{1}(0) = \max_{\tau \le 0} \sum_{\tau \le t \le 0} (S \max_{s \in [S]} \sum_{j} A_{j}(t) r_{j,s} - S \min_{s \in [S]} \sum_{m} U_{m} h_{m,s} + \bar{r}), \tag{20}$$

assuming that the process started at  $-\infty$ .

Let us define  $X_s(t)$  and  $Y_s(t)$  as follows:  $X_s(t) := \sum_j A_j(t) r_{j,s}$  and  $Y_s(t) := \sum_m U_m h_{m,s}$ . Then,

$$\tilde{Q}^{1}(0) = \max_{\tau \le 0} \sum_{\tau < t < 0} S(\max_{s} X_{s}(t) - \min_{s} Y_{s}(t) + \bar{r}).$$

Now, for any  $\theta > 0$ 

$$\Pr(\sum_{j} Q_{j,1} > \bar{r}q) \leq \Pr(\sum_{s} Q_{1}^{s} > q)$$

$$\leq \Pr(\tilde{Q}_{1}(0) > q)$$

$$= \Pr(\theta \tilde{Q}_{1}(0) > \theta q)$$

$$= \Pr(\exp(\theta \tilde{Q}_{1}(0)) > \exp(\theta q))$$

$$\leq \mathbb{E}[\exp(-\theta q)] \mathbb{E}[\exp(\theta \tilde{Q}_{1}(0))].$$
(21)

Now,

$$\mathbb{E}[\exp(\theta \tilde{Q}_{1}(0))] = \mathbb{E}[\exp(\theta S \left( \max_{\tau \leq 0} \sum_{\tau \leq t \leq 0} (\max_{s} X_{s}(t) - \min_{s} Y_{s}(t) + \bar{r}) \right))]$$

$$\leq \sum_{\tau \leq 0} \mathbb{E}[\exp(\theta S \sum_{\tau \leq t \leq 0} (\max_{s} X_{s}(t) - \min_{s} Y_{s}(t) + \bar{r}))], \tag{22}$$

where the inequality in (22) follows because for any random variables  $\{Z_j\}$ ,  $\exp(\theta \mathbb{Z}_j)$  are positive random variables and sum of positives are more than their maximum.

Next, we bound the term within the summation over  $\tau < 0$  in (22).

$$\mathbb{E}[\exp(\theta S \sum_{\tau \le t \le 0} (\max_s X_s(t) - \min_s Y_s(t) + \bar{r}))] \le \prod_{\tau \le t \le 0} \mathbb{E}[\exp(\theta (\max_s X_s(t) - \min_s Y_s(t) + \bar{r})))]. \tag{23}$$

Inequality in (23) follows because  $X_s(t)$ ,  $Y_s(t)$  are i.i.d. over time.

Next we bound the term within the product  $\prod_{\tau \le t \le 0}$  in (23),

$$\mathbb{E}\left[e^{\theta S(\max_s X_s(t) - \min_s Y_s(t) + \bar{r})}\right] \le \sum_{s,s'} \mathbb{E}\left[e^{\theta S(X_s(t) - Y_{s'}(t) + \bar{r})}\right]$$
(24)

Inequality (24) is due to the same reason as (22).

Let  $A_j(\theta) = \mathbb{E}\left[e^{\theta A_j(t)}\right]$  and  $\mathcal{U}_m(\theta) = \mathbb{E}\left[e^{\theta U_m(t)}\right]$  for  $j \in [N]$  and  $m \in [M]$ . For  $\theta \in \mathbb{R}$ , then,

$$\mathbb{E}\left[e^{\theta(X_{s}(t)-Y_{s'}(t))}\right] = \mathbb{E}\left[e^{\theta X_{s}(t)}\right] \mathbb{E}\left[e^{-\theta Y_{s'}(t)}\right] \\
= \mathbb{E}\left[e^{\theta \sum_{j} A_{j}(t)r_{j,s}+\bar{r}}\right] \mathbb{E}\left[e^{-\theta \sum_{i} U_{i}(t)h_{i,s}}\right] \\
= e^{\theta \bar{r}} \prod_{j} \mathbb{E}\left[e^{\theta A_{j}(t)r_{j,s}}\right] \prod_{i} \mathbb{E}\left[e^{-\theta U_{i}(t)h_{i,s'}}\right] \\
= e^{\theta \bar{r}} \prod_{j} \mathcal{A}_{j}(\theta r_{j,s}) \prod_{i} \mathcal{U}_{i}(-\theta h_{i,s'}) \\
= \exp\left(\theta \bar{r} + \sum_{j} \log \mathcal{A}_{j}(\theta r_{j,s}) + \sum_{i} \log \mathcal{U}_{i}(-\theta h_{i,s'})\right).$$
(25)

Note that as  $\lambda \in \alpha \mathcal{C}$ , by the definition of  $\mathcal{C}^O$ ,  $\sum_j \lambda_j r_{j,s} < \alpha \sum_m \mu_m h_{m,s}$  and by assumption  $|\sum_m \mu_m h_{m,s} - \sum_m \mu_m h_{m,s'}| \le \text{subpoly}(N)$  which is used in the following.

First consider the Gaussian-dominated case. Since the process variance is no more than mean and the moment generating function of the variance is upper-bounded by that of a zero-mean Gaussian:

$$\log \mathcal{A}_{j}(\theta r_{j,s}) \leq \lambda_{j} \theta r_{j,s} + \lambda_{j} \frac{(\theta r_{j,1,s})^{2}}{2}$$
$$\log \mathcal{U}_{i}(-\theta h_{i,s}) \leq -\mu_{j} \theta h_{i,s} + \mu_{j} \frac{(\theta h_{i,s})^{2}}{2}.$$

Note that for any two functions  $k_1x^2$  and  $k_2x$ ,  $\lim_{x\to 0}k_2x/k_1x^2=\infty$ , and hence for any  $\epsilon\in(0,1)$  there exists  $x^*>0$  such that for all  $x< x^*$ ,  $k_1x^2/k_2x<\epsilon$ . Hence for any  $\epsilon\in(0,1)$ , there exist  $\theta_{j,s}^*,\theta_{i,s}^*>0$ , for all i,j,s such that for all  $\theta<\theta^*=\min_{i,j,s}(\theta_{j,s}^*,\theta_{i,s}^*)$ ,

$$\log \mathcal{A}_{i}(\theta r_{i,1,s}) \le \lambda_{i} \theta^{*} r_{i,s} (1 + \epsilon) \tag{26}$$

$$\log \mathcal{U}_i(-\theta h_{i,s}) \le -\mu_i \theta h_{i,s}(1-\epsilon) \tag{27}$$

Note that since N, S, and M are finite and  $\theta_{j,s}^*, \theta_{i,s}^* > 0$ , for all  $i, j, s, \theta^* > 0$ . Moreover, note that  $\theta^*$  does not depend on  $\lambda, \mu$  since the ratio of the linear and quadratic terms in the log moment generating functions are independent of  $\lambda$  and  $\mu$ .

independent of  $\lambda$  and  $\mu$ . As  $e^{\theta}-1=\sum_{k=1}^{\infty}\frac{\theta^k}{k!}$ , for the Poisson-dominated case we have

$$\log \mathcal{A}_{j}(\theta r_{j,s}) \leq \lambda_{j} \sum_{k} \frac{(\theta r_{j,s})^{k}}{k!}$$
$$\log \mathcal{U}_{i}(-\theta h_{i,s}) \leq \mu_{j} \sum_{k} \frac{(-\theta h_{i,s})^{k}}{k!}$$

Again, by the same argument, we can have a  $\theta^*$  for which (26) and (27) are satisfied. As  $|\sum_i \mu_i h_{i,s} - \sum_i \mu_i h_{i,s'}| = o(N^\delta)$ , for all  $\delta > 0$ , and  $\sum_i \mu_i h_{i,s} = \Omega(N^c)$ , c > 0, for all  $\theta < \theta^*$  we have:

$$\mathbb{E}\left[e^{\theta(X_{s}(t)-Y_{s'}(t))}\right] \leq \exp\left(\theta^{*}\bar{r} + \sum_{j} \lambda_{j}\theta^{*}r_{j,s}(1+\epsilon) - \sum_{i} \mu_{i}\theta h_{i,s}(1-\epsilon) + \theta^{*}o(\mu_{i}\theta h_{i,s})\right) \\
\leq \exp\left(\theta\left(\bar{r} - \sum_{i} \mu_{i}h_{i,s}(\alpha - 2\epsilon)\right)\right).$$
(28)

Note (28) follows from the fact  $\lambda \in (1-\alpha)\mathcal{C}^{out}$ . As  $\epsilon > 0$  can be chosen arbitrarily small, we can have  $\alpha - 2\epsilon > 0$ . Since  $\sum_i \mu_i h_{i,s} > \sum_j (1-\alpha)\lambda_j r_{j,s}$  and  $\sum_j \lambda_j r_{j,1,s}$  scales with  $\lambda(N)$ , for sufficiently large  $\lambda_\alpha$  with  $\lambda_j \geq \lambda_\alpha$  for all j, we have  $\bar{r} - \sum_i \mu_i h_{i,s} (\alpha - \epsilon) \leq -\gamma \sum_i \mu_i h_{i,s} (\alpha - \epsilon)$ , for some  $\gamma > 0$ . Thus, we have for some  $\theta > 0$ ,

$$\mathbb{E}\left[e^{\theta S(X_s(t) - Y_{s'}(t))}\right] \le \exp\left(-\theta SK(N)\right),\tag{29}$$

where K(N) scales with N no slower than  $\sum_{s:r_{j,1,s}>0}\lambda_j(N)=\Omega(N^c),\ c>0.$  Thus,

$$\mathbb{E}\left[e^{\theta S(\max_s X_s(t) - \min_s Y_s(t))}\right] \le S^2 \exp\left(-\theta SK(N)\right).$$

Hence, from (22), (23), and (24) we have that

$$\begin{split} \mathbb{E}[\exp(\theta \sum_{s} \tilde{Q}^{s}(0))] &= \mathbb{E}[\exp(\theta S \tilde{Q}^{1}(0))] \\ &= \mathbb{E}[\exp(\theta \tilde{Q}(0))] \\ &\leq \sum_{\tau \leq 0} S^{|2\tau|} \exp(-\theta S K(N)|\tau|) \\ &\leq c', \end{split}$$

because  $S^2 < \exp(\theta^* SK(N))$  for all sufficiently large N.

Note that though we proved  $\mathbb{E}[\exp(\theta Q(t))] < c'$  for t = 0, this holds for any finite t with exactly the same proof. Hence,

$$\Pr(Q > c \log N) \le \Pr(\tilde{Q} > c \log N) \le c' e^{-c\theta^* \log N},$$

which gives the result for an appropriate choice of c.

## H. Proof of Theorem 14

To prove that any  $\lambda \in \mathcal{C}^L$  is stabilizable it is sufficient to invoke Thm. 15 whose proof is below. To show that  $\lambda^O \notin \mathcal{C}^I$  is not stabilizable, we take an approach similar to the proof of Thm. 5.

Note that the set  $\{\boldsymbol{\lambda}: \boldsymbol{\lambda} = \sum_{l} \boldsymbol{\lambda}^{l}, \boldsymbol{\lambda}^{l} \in \hat{\mathcal{C}}_{\Gamma}^{l}\}$  is convex. Also, if  $\boldsymbol{\lambda}' \leq \boldsymbol{\lambda}$  (component-wise) for some  $\boldsymbol{\lambda}$  belonging to the set, then  $\boldsymbol{\lambda}' \in \mathcal{C}^{I}$ . That is,  $\mathcal{C}^{I}$  is coordinate convex. Hence for any  $\boldsymbol{\lambda} \notin \text{closure of } \mathcal{C}^{L}$ , there exists a hyperplane  $h \geq 0$  that strictly separates it from  $\mathcal{C}^{L}$ , i.e., for any  $\{\boldsymbol{\lambda}^{l} \in \mathcal{C}^{l}\}$ , for some  $\epsilon > 0$ 

$$h^T \lambda^O \geq h^T \sum_{l} \boldsymbol{\lambda}^l + \epsilon$$

Note that at any epoch t, number of j jobs allocated  $\Delta_j(t) = \sum_l \Delta_j^l(t)$ , where  $\Delta_j^l(t)$  is the number of j jobs allocated to category l agents. Following similar steps as in the proof of Thm. 5, the result follows.

# I. Proof of Theorem 15

Let  $Q_{l,j,s}(t)$  be the number of unallocated (j,s) tasks in the lth pool and  $A_j(t)$  be the number of arrived jobs of type j. Let  $A_{l,j}(t)$  be the number of jobs that are sent to pool l and  $\sum_{l} A_{l,j} = A_j(t)$ . At any pool l MaxWeight is followed and  $D_{l,j,s}(t)$  is the number of allocated (j,s) tasks in pool l.

Consider a Lyapunov function  $L(\mathbf{Q}) = \sum_{l,j,s} Q_{l,j,s}^2$ . Then:

$$\mathbb{E}\left[L\left(\mathbf{Q}(t+1)\right) - L\left(\mathbf{Q}(t)\right)|\mathbf{Q}(t)\right] \\ \leq \mathbb{E}\left[\sum_{l,j,s} Q_{l,j,s}(t)A_{l,j}(t)|\mathbf{Q}(t)\right] - \mathbb{E}\left[\sum_{l,j,s} Q_{l,j,s}(t)\hat{\Delta}_{l,j,s}(t)|\mathbf{Q}(t)\right] + \mathbb{E}\left[\sum_{l,j,s} \left(A_{l,j}^{2}(t) + \hat{\Delta}_{l,j,s}^{2}\right)|\mathbf{Q}(t)\right]$$
(30)

The last term can be bounded by noting:

$$\sum_{l,j,s} \left( A_{l,j}^2(t) + \hat{\Delta}_{l,j,s}^2 \right) \le \left( \sum_{l,j,s} \left( A_{l,j}(t) + \hat{\Delta}_{l,j,s} \right) \right)^2$$

$$= \left( \sum_{j,s} A_j(t) + \sum_{l,j,s} \hat{\Delta}_{l,j,s} \right)^2$$

$$\le \left( \sum_{j,s} A_j(t) + \frac{1}{\min(r_{j,s} > 0)} \sum_{l,i,s} h_{i,s}^l U_{i,s}^l \right)^2$$

$$\le B' < \infty. \tag{32}$$

Note (31) follows similarly as (12), whereas (32) follows because the arrival and agent-availability processes have bounded second moments.

To bound the first term:

$$\mathbb{E}\left[\sum_{l,j,s} Q_{l,j,s}(t) A_{l,j}(t) | \mathbf{Q}(t)\right] = \mathbb{E}\left[\sum_{l,j} \left(\sum_{s} Q_{l,j,s}(t)\right) A_{l,j}(t) | \mathbf{Q}(t)\right]$$

$$\leq \mathbb{E}\left[\sum_{l,j,s} \frac{\lambda_j^l}{\lambda_j} A_j(t) Q_{l,j,s}(t) | \mathbf{Q}(t)\right]$$

$$\leq \sum_{l} \left(\sum_{j,s} Q_{l,j,s}(t) \lambda_j^l\right)$$
(34)

where (33) is because of the fact JLTT-MWTA sends all arrivals of type j to the pool l with minimum  $\sum_{s} Q_{l,j,s}(t)$ . On the other hand,

$$\mathbb{E}\left[\sum_{l,j,s} Q_{l,j,s}(t)\hat{\Delta}_{l,j,s}(t)|\mathbf{Q}(t)\right] = \sum_{l} \mathbb{E}\left[\sum_{j,s} Q_{l,j,s}(t)\hat{\Delta}_{l,j,s}(t)|\mathbf{Q}^{l}(t)\right],\tag{35}$$

because at epoch t each pool l runs MaxWeight based on only  $\mathbf{Q}_l(t)$  and  $\{\hat{\Delta}_{l,j,s}:j,s\}$  is independent of  $\{A_{l',j}(t),Q_{l',j,s}(t):l'\neq l\}$  given  $\mathbf{Q}_l(t)$ .

For every l, we can compute the difference between the lth term of (34) and that of (35), which is similar to the first term of (12). If  $\lambda + \epsilon \in \mathcal{C}^L$ , then  $\lambda^l + \frac{\epsilon}{L} \in \mathcal{C}^l$ , hence following the same steps as in the proof of Thm. 8 we have

$$\left(\sum_{j,s} Q_{l,j,s}(t)\lambda_j^l\right) - \mathbb{E}\left[\sum_{j,s} Q_{l,j,s}(t)\hat{\Delta}_{l,j,s}(t)|\mathbf{Q}^l(t)\right] \le -\frac{\epsilon}{L}\sum_{j,s} Q_{l,j,s}(t).$$

This in turn implies

$$\mathbb{E}\left[L\left(\mathbf{Q}(t+1)\right) - L\left(\mathbf{Q}(t)\right)|\mathbf{Q}(t)\right] \leq B' - \frac{\epsilon}{L} \sum_{l,j,s} Q_{l,j,s}(t).$$

Following similar steps as in proof of Thm. 8, we obtain  $\limsup_{t\to\infty}\sum_{l,j,s}\mathbb{E}\left[Q_{l,j,s}(t)\right]<\infty$ , which in turn implies that for all j,s,l,  $\limsup_{t\to\infty}\mathbb{E}\left[Q_{l,j,s}(t)\right]<\infty$ . This proves the theorem since for all  $j,Q_j(t)\leq\sum_{l,s}Q_{l,j,s}(t)$ .

# J. Proof of Theorem 16

It is sufficient to prove that  $\mathcal{C}^I\subseteq\mathcal{C}^O$ . Consider any  $\pmb{\lambda}\in\mathcal{C}^I$ , then by definition of  $\mathcal{C}^I$ ,  $\pmb{\lambda}=\sum_l\pmb{\lambda}^l$ , where for all  $l,\,\pmb{\lambda}^l\in\mathcal{C}^l$ . By the characterization of the outer region of single category systems  $\mathcal{C}^l\subseteq\mathcal{C}^{out}_l,\,\pmb{\lambda}^l\in\mathcal{C}^{out}_l$ . Thus  $\pmb{\lambda}=\sum_l\pmb{\lambda}^l$  for  $\pmb{\lambda}^l\in\mathcal{C}^{out}_l$ , for all l. Thus by definition of  $\mathcal{C}^O,\,\pmb{\lambda}\in\mathcal{C}^O$ , which completes the proof.

# K. Proof of Theorem 17

We find a high-probability bound on the number of unallocated jobs of type *j* and then bound the maximum number of jobs across type. To bound the number of unallocated jobs, we use stochastic domination based on the nature of the method of splitting job arrivals across different pools.

Consider the dynamics of  $Q_j^*(t) = \max_l Q_{l,j}(t)$ . Let  $A_{j,l}(t)$  be the number of jobs of type j that were directed to pool l. In the Improvised JSQ step, jobs are sent one-by-one with minimum backlog and hence, the queue  $l^*$  with  $Q_j^{l^*}(t) = Q_j^*(t)$  gets the minimum number of jobs. Since minimum is less than average,  $A_{j,l^*}(t) \leq \frac{A_j(t)}{L}$ . On the other hand, just before allocation at epoch t+1, the total number of j-jobs in  $l^*$  cannot be less than the number of jobs in any other l by more than 1. This is because of the Improvised JSQ which allocates jobs one-by-one to the lowest backlogged  $(N_i^l)$  queue at that time. Hence, we have

$$Q_j^*(t) + \lceil \frac{A_j(t)}{L} \rceil + 1 \ge Q_{j,l}(t) + A_{j,l}(t).$$

Given  $Q_{l,j}$ , for GreedyJob the number of jobs that can be allocated (assuming number of queued jobs to be infinite)  $\Delta_{l,j}$  is monotonic in  $\mathbf{U}^l$ , i.e., if  $\mathbf{U}^l \geq \mathbf{U'}^l$  (component-wise)  $\Delta_{l,j} \geq \Delta'_{l,j}$  for all j. This property will be useful below.

Consider the following dynamics  $Q^j(t), j \in [N]$ . Arrivals for each j are according to  $\lceil \frac{A_j(t)}{L} \rceil + 1$  and agent-availability is according to  $U^i(t) = \min_l U^l_i(t)$ . This is a single-category system and allocations in this system are according to GreedyJob. As  $U^i(t) \leq U^l_i(t)$  for all i, l, the number of allocations (assuming queues to be infinite) satisfies  $\Delta^j(t) \leq \Delta_{l,j}(t)$ . This implies that for each type j  $Q^j(t)$  dominates  $Q^*_j(t)$  as the first queue at any epoch has more number of jobs to be allocated and less number of possible allocations (as  $\Delta^j$  is smaller). Thus, it is sufficient to bound  $Q^j(t)$ .

For this we proceed along the lines of the proof of Thm. 13, replacing arrivals by  $A^j(t) = \lceil \frac{A_j(t)}{L} \rceil + 1$  and agent-availability by  $U^i(t) = \min_l U^l_i(t)$ .

Hence, for each skill s, the queue of unallocated skill-hours  $H^s(t)$  is the arrival  $X^s(t) = \sum_j A^j(t) r_{j,s}$  and the possible amount that can be drained is  $Y^s(t) = \sum_j U^i(t) h_{i,s}$ . Hence,

$$Q^{s}(t+1) = |Q^{s}(t) + X^{s}(t) - Y^{s}(t) + \bar{r}|^{+}.$$

We can follow similar steps by noting that for  $\theta > 0$ :

$$\mathbb{E}\left[e^{-\theta Y^{s}}\right] = \mathbb{E}\left[e^{-\theta \sum_{i} \min_{l} U_{i}^{l}(t)h_{i,s}}\right]$$

$$= \mathbb{E}\left[e^{-\theta \min_{l} \sum_{i} U_{i}^{l}(t)h_{i,s}}\right]$$

$$= \mathbb{E}\left[\max_{l} e^{-\theta \sum_{i} U_{i}^{l}(t)h_{i,s}}\right]$$

$$\leq \sum_{l} \mathbb{E}\left[e^{-\theta \sum_{i} U_{i}^{l}(t)h_{i,s}}\right].$$
(36)

On the other hand,

$$\mathbb{E}\left[e^{\theta X^s}\right] \leq e^{\theta(\frac{1}{L}+1)\bar{r}} \prod_{j} \mathbb{E}\left[e^{\frac{\theta r_{j,s}}{L}A_j(t)}\right].$$

Making the assumption on agent arrival rates,

$$\sum_{l} \mathbb{E}\left[e^{-\theta \sum_{i} U_{i}^{l}(t)h_{i,s}}\right] \leq e^{\log L + O\left(\text{subpoly}(N)\right)} \mathbb{E}\left[e^{-\theta \sum_{i} U_{i}^{1}(t)h_{i,s}}\right]$$
(37)

Since  $\max_{l,l'} |\mu_i^l - \mu_i^{l'}| = \text{subpoly}(N)$  for all i,  $\lambda \in (1 - \alpha)\mathcal{C}^O$  implies that  $\frac{\lambda}{L} - \text{subpoly}(N) \in \mathcal{C}^{out}$ . Thus, we can use the same steps as we did for single-category systems.

For Gaussian-dominated as well as Poisson-dominated cases, following similar steps we can obtain that for a  $\theta^* > 0$  (independent of  $\lambda$ ) and sufficiently large  $\lambda$ ,

$$\mathbb{E}\left[e^{\theta(X^s-Y^s)}\right] \leq e^{-\theta^*K(\pmb{\lambda}) + \log L + O\left(\mathsf{subpoly}(N)\right)}$$

Note that  $K(\lambda)$  increases as  $\Omega(\min_i \lambda_i) = \Omega(N^c)$  for some c > 0, L = O(1), hence there exists  $\lambda$  sufficiently large such that  $-\theta^*K(\lambda) + \log L + O\left(\text{subpoly}(N)\right)$  is strictly negative and  $\mathbb{E}\left[e^{\theta(X^s - Y^s)}\right] < 1$ . The rest follows similarly as the proof of Thm. 11.

# L. Proof of Theorem 18

We first consider FD, FND, and IND systems.

Let  $\hat{A}_j(t)$  be the number of accepted jobs of type j between starts of epochs t-1 and t. Let  $\hat{\Delta}_{j,s}(t)$  be the number of allocated (j,s) tasks by the MaxWeight part of MWTA before the execution of Task Allocation. Let  $\hat{D}_{j,s}(t)$  be the number of allocated (j,s) tasks at allocation epoch t at the end of Task Allocation. Then:

$$\sum_{j,s:r_{j,s}>0} \left( \tilde{Q}_{j,s}^{2}(t+1) - \tilde{Q}_{j,s}^{2}(t) \right) = \sum_{j,s:r_{j,s}>0} \left( (\tilde{Q}_{j,s}(t) + \tilde{A}_{j}(t) - \hat{D}_{j,s}(t))^{2} - \tilde{Q}_{j,s}^{2}(t) \right) \\
\leq \sum_{j,s:r_{j,s}>0} \left( (\tilde{Q}_{j,s}(t) + \tilde{A}_{j}(t) - \hat{\Delta}_{j,s}(t))^{2} - \tilde{Q}_{j,s}^{2}(t) \right) \\
= \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \hat{\Delta}_{j,s}(t) \right) + \sum_{j,s:r_{j,s}>0} \left( \tilde{A}_{j}^{2}(t) + \Delta_{j,s}^{2}(t) \right) \\
= \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \hat{\Delta}_{j,s}(t) \right) + \sum_{j,s:r_{j,s}>0} \left( \tilde{A}_{j}^{2}(t) + \Delta_{j,s}^{2}(t) \right) \\
= \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \hat{\Delta}_{j,s}(t) \right) + \sum_{j,s:r_{j,s}>0} \left( \tilde{A}_{j}^{2}(t) + \tilde{A}_{j,s}(t) \right) \\
= \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \hat{\Delta}_{j,s}(t) \right) + \sum_{j,s:r_{j,s}>0} \left( \tilde{A}_{j}^{2}(t) + \tilde{A}_{j,s}(t) \right) \\
= \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \hat{\Delta}_{j,s}(t) \right) + \sum_{j,s:r_{j,s}>0} \left( \tilde{A}_{j}^{2}(t) + \tilde{A}_{j,s}(t) \right) \\
= \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \tilde{A}_{j,s}(t) \right) + \sum_{j,s:r_{j,s}>0} \left( \tilde{A}_{j}^{2}(t) + \tilde{A}_{j,s}(t) \right) \\
= \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \tilde{A}_{j,s}(t) \right) + \sum_{j,s:r_{j,s}>0} \left( \tilde{A}_{j}^{2}(t) + \tilde{A}_{j,s}(t) \right) \\
= \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \tilde{A}_{j,s}(t) \right) + \sum_{j,s:r_{j,s}>0} \left( \tilde{A}_{j}^{2}(t) + \tilde{A}_{j,s}(t) \right) \\
= \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \tilde{A}_{j,s}(t) \right) + \sum_{j,s:r_{j,s}>0} \left( \tilde{A}_{j}(t) - \tilde{A}_{j,s}(t) \right) \\
= \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \tilde{A}_{j,s}(t) \right) + \sum_{j,s:r_{j,s}>0} \left( \tilde{A}_{j}(t) - \tilde{A}_{j,s}(t) \right) \\
= \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \tilde{A}_{j,s}(t) \right) + \sum_{j,s:r_{j,s}>0} Q_{j,s}(t) \left( \tilde{A}_{j}(t) - \tilde{A}_{j,s}(t) \right)$$

Expectation of the second summation (conditioned on  $\mathbf{Q}(t)$ ) can be bounded by noting that  $\tilde{A}_{j}^{2}(t) \leq A_{j}^{2}(t)$  and the fact that the arrival processes have bounded second moment. The value  $\mathbb{E}\left[\sum_{j,s:r_{j,s}>0}\Delta_{j,s}^{2}(t)\right]$  can be bounded similarly as in the proof of Thm. 8. Hence, we consider the expectation of the second term to be bounded by B independent of  $\mathbf{Q}$ .

$$\begin{split} &\mathbb{E}\left[\sum_{j,s:r_{j,s}>0}\left(\tilde{Q}_{j,s}^{2}(t+1)-\tilde{Q}_{j,s}^{2}(t)\right)|\mathbf{Q}(t)\right] \\ &\leq B+\mathbb{E}\left[\sum_{j,s:r_{j,s}>0}Q_{j,s}(t)\left(\tilde{A}_{j}(t)-\hat{\Delta}_{j,s}(t)\right)|\mathbf{Q}(t)\right] \\ &\leq B+\mathbb{E}\left[\mathbb{E}\left[\sum_{j,s:r_{j,s}>0}Q_{j,s}(t)\left((1-\beta(t))A_{j}(t)-\hat{\Delta}_{j,s}(t)\right)|\mathbf{Q}(t),\mathbf{A}(t)\right]\mathbf{Q}(t)\right] \\ &\leq B+\mathbb{E}\left[\mathbb{E}\left[\frac{\beta(t)\sum_{j}A_{j}(t)}{\nu}+\sum_{j,s:r_{j,s}>0}Q_{j,s}(t)\left((1-\beta(t))A_{j}(t)-\hat{\Delta}_{j,s}(t)\right)-\frac{\beta(t)\sum_{j}A_{j}(t)}{\nu}|\mathbf{Q}(t),\mathbf{A}(t)\right]|\mathbf{Q}(t)\right] \\ &\leq B+\mathbb{E}\left[\mathbb{E}\left[\frac{\beta(t)\sum_{j}A_{j}(t)}{\nu}+\sum_{j,s:r_{j,s}>0}Q_{j,s}(t)\left((1-\beta(t))\mathbb{E}[A_{j}(t)]-\hat{\Delta}_{j,s}(t)\right)-\frac{\beta(t)\sum_{j}A_{j}(t)}{\nu}|\mathbf{Q}(t),\mathbf{A}(t)\right]|\mathbf{Q}(t)\right] \\ &\leq B+\mathbb{E}\left[\mathbb{E}\left[\frac{\beta(t)\sum_{j}A_{j}(t)}{\nu}+\sum_{j,s:r_{j,s}>0}Q_{j,s}(t)\left((1-\beta(t))A_{j}(t)-\hat{\Delta}_{j,s}(t)\right)-\frac{\beta(t)\sum_{j}A_{j}(t)}{\nu}|\mathbf{Q}(t),\mathbf{A}(t)\right]|\mathbf{Q}(t)\right] \\ &\leq B+\mathbb{E}\left[\mathbb{E}\left[\frac{\beta^{*}\sum_{j}A_{j}(t)}{\nu}+\sum_{j,s:r_{j,s}>0}Q_{j,s}(t)\left((1-\beta^{*})A_{j}(t)-\hat{\Delta}_{j,s}(t)\right)-\frac{\beta(t)\sum_{j}A_{j}(t)}{\nu}|\mathbf{Q}(t),\mathbf{A}(t)\right]|\mathbf{Q}(t)\right] \\ &\leq B+\frac{1}{\nu}\mathbb{E}[(\beta^{*}-\beta(t))\sum_{j}A_{j}(t)]+\mathbb{E}\left[\sum_{j,s:r_{j,s}>0}Q_{j,s}(t)\left((1-\beta^{*})A_{j}(t)-\hat{\Delta}_{j,s}(t)\right)|\mathbf{Q}(t)\right] \\ &\leq B+\frac{1}{\nu}\mathbb{E$$

As  $(1 - \beta^*)\lambda + \epsilon \mathbf{1} \in \mathcal{C}$ , following the same steps as in the proof of the Thm. 8:

$$\mathbb{E}\left[\sum_{j,s:r_{j,s}>0}Q_{j,s}(t)\left((1-\beta^*)\lambda_j(t)-\hat{\Delta}_{j,s}(t)\right)|\mathbf{Q}(t)\right] \leq -\epsilon \sum_{j,s:r_{j,s}>0}Q_{j,s}(t)$$

As  $\beta^* - \beta(t) \le 1$ , we have

$$\mathbb{E}\left[\sum_{j,s:r_{j,s}>0} \left(\tilde{Q}_{j,s}^2(t+1) - \tilde{Q}_{j,s}^2(t)\right)\right] \le B + \frac{1}{\nu} \sum_{j} \lambda_j - \epsilon \mathbb{E}\left[\sum_{j,s:r_{j,s}>0} Q_{j,s}(t)\right].$$

Following again the same steps we show that  $\limsup_{t\to\infty}\mathbb{E}\left[\sum_{j,s:r_{j,s}>0}Q_{j,s}(t)\right]<\infty$  which implies the crowd system is stable. Hence, we can assume that  $\mathbb{E}\left[\sum_{j,s:r_{j,s}>0}Q_{j,s}(t)\right]< C$  for some  $C<\infty$ .

Thus we can write

$$\mathbb{E}\left[\sum_{j,s:r_{j,s}>0} \left(\tilde{Q}_{j,s}^{2}(T) - \tilde{Q}_{j,s}^{2}(0)\right)\right] \leq BT + \frac{1}{\nu} \sum_{t=1}^{T} \mathbb{E}[(\beta^{*} - \beta(t)) \sum_{j} A_{j}(t)],$$

which in turn implies

$$(1 - \beta^*) \sum_{j} \lambda_j - \frac{1}{T} \sum_{t=1}^{T} (1 - \beta(t)) \sum_{j} A_j(t) \le \nu B + \frac{\nu}{T} \sum_{j,s:r_{j,s} > 0} \tilde{Q}_{j,s}^2(0)$$

Since B is a constant depending on arrival and availability statistics,  $\nu$  and T can be chosen to be small and large respectively to ensure that the left side is arbitrarily small. The desired result follows by noting that

$$\mathbb{E}\left[\sum_{t=1}^{T} (1 - \beta(t)) \sum_{j} A_{j}(t)\right] = \mathbb{E}\left[\sum_{t=1}^{T} \sum_{j} \tilde{A}_{j}(t)\right].$$

Also note that since the only requirement is the independence of A(t) and U(t) across time, the proof directly extends to settings with non-stationary arrival and availability processes.

Now consider the ID setting and use the Lyapunov function  $\sum_{l,j,s} Q_{l,j,s}^2$ . Similar to before, the Lyapunov drift can be bounded by

$$B + \mathbb{E}\left[\sum_{l,j,s:r_{j,s}>0} Q_{l,j,s}(t) \left(\tilde{A}_{l,j}(t) - \hat{\Delta}_{l,j,s}(t)\right) | \mathbf{Q}(t)\right].$$

Note that  $\sum_{l,j,s:r_{j,s}>0} Q_{l,j,s}(t) \tilde{A}_{l,j}(t)$  is equal to

$$\sum_{\substack{l,i,s:r_{l,s}>0}} \min_{l} \left( \sum_{s} Q_{l,j,s}(t) \right) \tilde{A}_{j}(t),$$

as  $\tilde{A}_j(t) = \sum_l \tilde{A}_{l,j}(t)$  and JLTT ensures that the jobs are sent to the category with  $\min_l (\sum_s Q_{l,j,s}(t))$ . So we have

$$\begin{split} & \mathbb{E}\left[\sum_{l,j,s:r_{j,s}>0}Q_{l,j,s}(t)\left(\tilde{A}_{l,j}(t)-\hat{\Delta}_{l,j,s}(t)\right)|\mathbf{Q}(t)\right] \\ & = \mathbb{E}\left[\sum_{l,j:r_{j,s}>0}\min_{l}\left(\sum_{s}Q_{l,j,s}(t)\right)\tilde{A}_{j}(t) - \sum_{l,j,s:r_{j,s}>0}Q_{l,j,s}(t)\hat{\Delta}_{l,j,s}(t)|\mathbf{Q}(t)\right] \\ & \leq \mathbb{E}\left[\sum_{l,j:r_{j,s}>0}Q_{l,j,s}(t)\frac{\lambda_{j}^{l}}{\lambda_{j}}\tilde{A}_{j}(t) - \sum_{j,s:r_{j,s}>0}Q_{l,j,s}(t)\hat{\Delta}_{l,j,s}(t)|\mathbf{Q}(t)\right], \end{split}$$

where  $(1 - \beta^*)\lambda = \sum_l (1 - \beta^*)\lambda^l$  for  $\lambda^l + \epsilon \mathbf{1} \in \mathcal{C}^l$ . The remainder of the proof is similar to the approach for the FD/FND/IND settings, i.e., the MWTA proof for  $(1 - \beta^*)\lambda^l + \epsilon \mathbf{1} \in \mathcal{C}^l$  for each l.