Social Distancing via Social Scheduling

Deepesh Kumar Lall[*],[1] **Garima Shakya**[*],[2] **Swaprava Nath**[3]

[1] Oracle, India
[2] Kyushu University, Japan
[3]Indian Institute of Technology Bombay, India
deepesh.l.lall@oracle.com, garima@inf.kyushu-u.ac.jp, swaprava@cse.iitb.ac.in

Abstract

Motivated by the need for *social distancing* during a pandemic, we consider an approach to schedule the visitors of a facility (e.g., a general store). Our algorithms take input from the citizens and schedule the store's discrete time-slots based on their importance to visit the facility. We consider *indivisible* customer job requests that take single or multiple slots to complete. The salient properties of our approach are: it (a) ensures social distancing by ensuring a maximum population in a given time-slot at the facility, (b) prioritizes individuals based on the importance of the jobs, (c) maintains truthfulness of the reported importance by adding a *cooling-off* period after their allocated time-slot, during which the individual cannot re-access the same facility, (d) guarantees voluntary participation of the citizens, and yet (e) is computationally tractable. The mechanisms we propose are prior-free. The problem is NP-complete for indivisible *multi-slot* jobs, and we provide a polynomial-time mechanism that is truthful, individually rational, and approximately optimal. Experiments with data collected from a store show that visitors with more important (single-slot) jobs are allocated more preferred slots, which comes at the cost of a longer cooling-off period and significantly reduces social congestion. For the multi-slot jobs, our mechanism yields reasonable approximation while reducing the computation time significantly. While our solutions are primarily motivated by the ongoing raging pandemic, our formulation naturally applies to a broad range of scheduling settings.

Introduction

Pandemics show the limits of pharmaceutical interventions (e.g., vaccines). Infectious diseases have appeared multiple times in the history of the human race (Spanish flu, Ebola, SARS, COVID-19, etc.) and vaccine-development took different approaches. However, the unique first defense had always been a non-pharmaceutical intervention called *social distancing*, a term that has been added to all major English dictionaries in 2020. From the times of the Spanish flu (1918) (Hatchett, Mecher, and Lipsitch 2007) to the recent COVID-19 (Wilder-Smith and Freedman 2020) it has been proved to be the most effective early solution.

The method of social distancing, however, has evolved. From mass exodus of population from the afflicted areas or forcing ships to anchor for months before entering a port in the early 20th century, we can now leverage the communication and AI technologies to efficiently execute social distancing without disrupting human habitation or occupation. What we need is to *socially schedule* these individuals to visit the public facilities and prevent them from overcrowding. This is the approach we consider in this paper.

In our setting, each customer has an infinite queue of jobs that have different importances and lengths (both are privately known only to the customer). However, the customers are *myopic*, i.e., worry only about the last unprocessed job.[1] They experience a better value if the job is assigned their preferred slots, but also a *disutility* to wait before submitting their next job for allocation. All jobs are *indivisible*, i.e., has to be completed once started. In this paper, we consider *two* settings: (i) all jobs are of single time-slot length, (ii) different jobs are of different integral time-slot lengths.

Though cast in the context of social scheduling for pandemics, a similar problem arises in general scheduling settings, e.g., scheduling traffic in freeways or multi-ownership computational jobs in a single-core processor. Since all such settings have multiple agents competing for a common resource and the importance of the jobs are private, the solutions involving truthful revelation in a computationally tractable manner also apply to those settings.

This paper introduces a novel approach to pandemic containment using mechanism design that reduces the congestion in facilities, satisfies various desirable theoretical properties, and exhibits fair performance in practice. The following section details the contributions of this paper.

Brief Problem Description and Contributions

The opening hours of a facility are divided into *periods* (e.g., a day), each of which has multiple *slots* (e.g., every hour when it is open). The customers have an unlimited number of outstanding jobs to be processed in a sequence at the facility, and they report the valuations of the immediate un-

[*]These authors contributed equally.

[1]A large number of studies in behavioral science point to such myopic behavior (Langer and Weber 2005).

processed job.[2] A valuation v_{ij} denotes agent i's importance for that job if it starts in slot j. Since this information is private to agent i, a mechanism needs to elicit this truthfully. In a setting where the agents' preferences are private, if the mechanism has no additional structures (e.g., transfer of individual payoff), only *dictatorial* mechanisms (where a pre-selected agent's favorite outcome is always selected) are truthful (Roberts 1979, Thm 7.2). Therefore, the use of transfers in some form is inevitable to ensure truthfulness of the agents. However, for pandemic containment, the use of money for scheduling citizens is unethical and illegal in certain countries. Hence, we use *time-delay* as a replacement of money. Waiting time is often seen as a resource that individuals agree to trade with (Leclerc, Schmitt, and Dube 1995). Our scheduling approach will work in all places where payment can be replaced with a time-delay. Quite naturally, an agent prefers to have a more valuable slot assigned to her with less time-delay. We model the agents' payoffs using the well-known quasi-linear payoff model (Shoham and Leyton-Brown 2008, Chap 10). This competitive scenario induces a *game* where agents' actions are to report the valuations. The agents may *overstate* (or *understate*) their actual valuations.

The contributions of this paper can be summarized into the following *four* major points:

- ▷ We show that the problem of maximizing the *social welfare* (sum of the agents' valuations) of the slot allocation is computationally easy to solve for jobs with single-slot length (Theorems 6 and 7) despite it being a combinatorial optimization problem.
- ▷ The single-slot case has the advantage that the delay (cooling-off time) can be calculated via the Vickrey-Clarke-Groves (VCG) mechanism (Vickrey 1961; Clarke 1971; Groves 1973) that ensures truthfulness and participation of the agents.
- ▷ Our main contribution is in the multi-slot jobs. We show that the welfare maximizing allocation of multi-slot jobs is computationally hard (Theorem 1). We propose a polynomial time mechanism (Theorem 4) which ensures participation (Theorem 2), truthfulness (Theorem 3), and is approximately optimal (Theorem 5).
- ▷ Our real and synthetic data experiments show that visitors with more important jobs are allocated more preferred slots, which comes at the cost of a longer delay to re-access the store. We show that social distancing is significantly improved using users' visit data from a store. For the multi-slot jobs, our approximately optimal mechanism provides a reasonable approximation at a much reduced computational cost in practice.

The mechanisms presented in this paper are *prior-free*, i.e., they do not depend on the probabilistic information of the valuations.

Related Work

The literature on pandemic control has widely documented the quantitative benefits of social distancing (Glass et al. 2006; Thunström et al. 2020; Fong et al. 2020). Studies related to the behaviour of individuals during pandemic show that, despite the infection probability not decreasing, during the equilibrium social distancing phase, individuals gradually reduce their social distancing efforts (Toxvaerd 2020; Cavallo 2021).

In a slightly different strand of literature, the problem of resource allocation with monetary transfers has been addressed to ensure truthfulness, e.g., in the context of job-shop scheduling (Hajiaghayi 2005), which is also close to our work. Lavi and Nisan (2004) study online supply curves based auction of *identical divisible goods* that ensures truthfulness. In this paper, we consider an offline allocation problem but a comparatively more complex one (multiple resources and indivisible tasks of different length). Chen et al. (2016) propose a truthful approximate mechanism for online allocation of job to machines where the job can resume or restart once preempted. We provide a comparatively better approximation ratio for efficiency, albeit in an offline setting. The other related line of work involves designing incentives in queueing problems with *specific cost structures* that aim to find an efficient allocation truthfully and also ensures budget balance (Mitra 2001; Bloch 2017; Ghosh, Long, and Mitra 2020), while our model can admit costs of *any* structure.

In the context of job scheduling without money, Koutsoupias (2014) studies the allocation of independent tasks to machines. Every machine reports the time it takes to execute each task and the mechanism provides an approximation to the minimum *makespan* in a truthful manner without money for one task—which can be repeated for multiple tasks. In this paper, we maximize the *sum of the visitors' valuations* which are independent of the length of the job and provide an approximate mechanism for multiple tasks maintaining the slot-capacity. Braverman et al. (2016) study a similar problem of the allocation of medical treatments at hospitals that have differential costs to patients and the patients value the hospitals differently. The waiting time before being admitted to the hospital helps to get a stable matching. However, the value of the agents do not change over slots and hence is different from our setting.

Basic Setup and Single-slot Jobs

Define $N := \{1, \ldots, n\}$ to be the set of agents that are trying to access a facility $\mathcal{F}$. Time is divided into *periods*, and each period is further divided into *slots*. The set of slots is denoted by $M := \{1, \ldots, m\}$. Every slot has a maximum capacity of k, which is decided by the region's social distancing norm based on the size of the facility.[3] A *central planner* (e.g., an AI app) allocates these slots to the agents, maintaining the capacity constraint. Every agent i has a *cardinal* preference $v_{ij} \in \mathbb{R}_{\geqslant 0}$ (called the agent's *valuation*) if her immediate unprocessed job is allocated slot j. The valuation implicitly reflects the importance of visiting the facility for that agent. The valuation vector of i is represented by $v_i = (v_{ij}, j \in M) \in \mathbb{R}^m_{\geqslant 0}$. In this paper, we consider differ-

[2]A typical shopper knows that she needs to visit a store many times but precisely knows the importance of the immediate visit.

[3]The analysis and results will follow even if the capacity k_j varies with the slots $j \in M$.

ent facilities independently. The joint facility-slot allocation problem can be modeled as a similar problem with the additional constraint that the same slot cannot be allocated to the same agent at different facilities. We leave the detailed analysis for it as future work.

The planner decides the allocation which can be represented as a matrix $A = [a_{ij}]$, where $a_{ij} = 1$, if agent i is allotted slot j, and zero otherwise. We assume that every agent can be assigned at most one slot in a period, and the total number of agents assigned to each slot does not exceed k. We denote the slot assigned to i by a_i^*. The planner also decides a delay $d = (d_i, i \in N)$, where d_i is the time-delay (in the same unit as the valuation) of agent i before which she cannot make another request to the system. The net payoff of an agent is assumed to follow a standard *quasi-linear form* (Shoham and Leyton-Brown 2008), which implies that every agent wants a more valued slot to be assigned to her and also wants to wait less.

$$u_i((A, d), v_i) = v_i(A) - d_i, \text{ where } v_i(A) = v_{ia_i^*}. \quad (1)$$

Denote the set of all allocations by $\mathcal{A}$. The delays $d_i \in \mathbb{R}_{\geqslant 0}, \forall i \in N$. The planner does not know the valuations of the agents. Therefore he needs the agents to report their valuations to decide the allocation and the delay. This leaves the opportunity for an agent to misrepresent her true valuation. To distinguish, we use v_{ij} for the true valuation and $\hat{v}_{ij}$ for reported valuations. In the first part of this paper, we will consider single-slot jobs and use the shorthand $v = (v_i)_{i \in N}$ to denote the true valuation profile represented as an $m \times n$ real matrix, and $\hat{v}$ to denote the reported valuation profile. The notation v_{-i} denotes the valuation profile of the agents except i. The decision problem of the planner is, therefore, formally captured by the following function.

DEFINITION 1 (Social Scheduling Function (SSF)). *A* social scheduling function (SSF) *is a mapping $f : \mathbb{R}^{m \times n} \to \mathcal{A} \times \mathbb{R}^n$ that maps the reported valuations to an allocation and delay for every agent. Hence, $f(\hat{v}) = (A(\hat{v}), d(\hat{v}))$, where A is the allocation and d is the delay function.*[4]

Preliminary Definitions

In this section, we formally define a few desirable properties that a social scheduling function should satisfy. The properties address the issues of prioritization, truthfulness, voluntary participation, and computational complexity.

The first property ensures that the allocation is *efficient* in each period, i.e., it maximizes the sum of the valuations of all the agents.

DEFINITION 2 (Efficient Per Period (EPP)). *An SSF f is* efficient per period (EPP) *if at every period, it chooses an allocation A^* that maximizes the sum of the valuations of all the agents. Formally, if $f(\cdot) = (A^*(\cdot), d(\cdot))$, then*

$$A^*(v) \in \operatorname*{argmax}_{A \in \mathcal{A}} \sum_{i \in N} \sum_{j \in M} v_{ij} a_{ij}. \quad (2)$$

[4]We overload the notation A and d to denote both functions and values of those functions, since their use will be clear from the context.

However, since the planner can only access the reported values $\hat{v}_i$'s, which can be different from the true v_i's, it is necessary that the reported values are indeed the true values. The following property ensures that the agents are incentivized to 'truthfully' reveal this information *irrespective of the reports of the other agents*.

DEFINITION 3 (Per Period Dominant Strategy Truthful). *An SSF $f(\cdot) = (A(\cdot), d(\cdot))$ is* truthful in dominant strategies per period *if for every $v_i, \hat{v}_i, \hat{v}_{-i}$, and $i \in N$*

$$v_i(A(v_i, \hat{v}_{-i})) - d_i(v_i, \hat{v}_{-i}) \geqslant v_i(A(\hat{v}_i, \hat{v}_{-i})) - d_i(\hat{v}_i, \hat{v}_{-i}).$$

The next property ensures that it is always weakly beneficial for every rational agent to participate in such a mechanism.

DEFINITION 4 (Individual Rationality). *An SSF $f(\cdot) = (A(\cdot), d(\cdot))$ is* individually rational *if for every v, and $i \in N$*

$$v_i(A(v)) - d_i(v) \geqslant 0.$$

Large facilities that have a large number of high-capacity slots lead to an exponential increase in the size of the set $\mathcal{A}$. This largeness of $\mathcal{A}$ makes it challenging to find a solution quickly. In a practical setting, where the allocations and delays need to be decided before every period, it is desirable to have an SSF that is computable in a time polynomial in n and m so that it finishes the computation in a time negligible to the time duration of the period. We consider algorithms that are *strongly polynomial* (Grötschel, Lovász, and Schrijver 1993). An SSF is strongly polynomial-time computable if there exists an algorithm that computes it in a time strongly polynomial in n and m, irrespective of the size of the actual data, such as the value of the v_is or k.

Periodic Mechanisms

We consider mechanisms that run at every period of this social scheduling problem. The agents report their valuations at the beginning of every period. The planner decides the schedules and delays.[5] Since the agents have the opportunity to overstate their importance to get a better slot allotted to them, our approach that uses the ideas of mechanism design (Börgers 2015) to this social scheduling problem is useful. We use the delay as a surrogate for transferable utility among the agents to satisfy several desirable properties.

For the single-slot job setup, the delays of agents are computed via the standard VCG payment rule and we call the allocation and delay together as the mechanism **VCG-T** (**VCG** with **T**ime delays). **VCG-T** mechanism is as follows.

Description of VCG-T. The SSF needs to decide on the allocation A and the delay d. **VCG-T** computes the alloca-

[5]For mechanisms that consider the dynamic extension of such allocation problems with finite or infinite horizon (Bergemann and Välimäki 2010, e.g.,) (a) the designer needs to know the transition probabilities, (b) equilibrium guarantees are weaker, and (c) are computationally expensive. These factors made us restrict our attention to periodic mechanisms.

tion as follows.

$$\begin{aligned} &\underset{A}{\operatorname{argmax}} \sum_{j \in M} \sum_{i \in N} v_{ij} a_{ij} \\ &\text{s.t.} \sum_{j \in M} a_{ij} \leqslant 1, \; \forall i \in N; \; \sum_{i \in N} a_{ij} \leqslant k, \; \forall j \in M \\ &a_{ij} \geqslant 0, \; \forall i \in N, j \in M. \end{aligned} \tag{3}$$

This is an LP relaxation of the actual allocation problem, which allows a_{ij}s to be only in $\{0,1\}$. We will show that this is without loss of optimality since the solution to LP (3) will always be integral and will coincide with the solution of the corresponding IP.

The delays of agents are computed via the standard VCG payment rule. Denote the optimal allocation of LP (3) by $A^*(v)$. Also, denote the allocation given by LP (3) when agent i is removed from the system by $A^*_{-i}(v_{-i})$. For agent i, the delay is given by,

$$d_i := \sum_{\ell \in N \setminus \{i\}} v_\ell(A^*_{-i}) - \sum_{\ell \in N \setminus \{i\}} v_\ell(A^*). \tag{4}$$

The mechanism in every period is described in Algorithm 1.

Algorithm 1: **VCG-T** in every period

```
1: Input: for every agent $i \in N$, the value $\hat{v}_i$
2: compute $A^*(\hat{v})$ (Equation (3)) as the allocation
3: charge a delay $d_i(\hat{v})$ (Equation (4)) to every $i \in N$
   for which they cannot access the scheduling mechanism
   again
4: Output: $A^*(\hat{v})$ and $d(\hat{v})$
```

We show that the **VCG-T** mechanism for single-slot job is per period dominant strategy truthful, individually rational, and runs in strongly polynomial time. Due to the page limitation, we put the results in the appendices available in the supplementary material.

The main contribution of this paper is to schedule the multi-slot jobs, that are relatively difficult to schedule. We present the our results for multi-slot jobs in the next section.

Multi-slot Jobs

In this section, we consider jobs with different lengths, i.e., for agent i, the job may be of length $l_i \geqslant 1$. Since the job is *indivisible*, the entire length l_i of the job requires contiguous slots for execution within the period. For example, an individual may visit a facility (e.g., a shopping mall) for a quick shopping, which may take a shorter duration, or for dining, which may take longer. However, all these jobs are indivisible, and the allocation needs to provide contiguous time-slots to that agent. The agents report the valuations and lengths of their jobs. We show that the optimal allocation problem in such a setting can be computationally intractable. The notation is mildly updated as follows to accommodate the multi-slot jobs.

Each agent i gets a valuation v_{ij} for her last unprocessed job if her job begins at slot j, and has a length l_i. The value of the job is zero if (a) it starts at any of the last $(l_i - 1)$ time-slots of the period (since it cannot finish within the period), and (b) if the job is unallocated.

A matrix V consists of the agents' reported valuations, and L consists of the lengths of agents' jobs. Allocation is given by the matrix $\mathbf{A} = [\mathfrak{a}_{ij}]$, where $\mathfrak{a}_{ij} = 1$ if agent i's job starts at slot j, else $\mathfrak{a}_{ij} = 0$, and $\mathfrak{a}_i$ represents the slot allocation vector for agent i. Keeping all other notations as before, we define the **MIA** problem as follows.

DEFINITION **5** (Multi-slot Indivisible jobs Allocation problem (**MIA**)). *: Given (N, M, V, L, k), find an allocation $\mathbf{A}$, such that $\sum_{i \in N} \sum_{j \in M} v_{ij}(\mathfrak{a}_{ij})$ is maximum, subject to the constraints that the total number of jobs allocated in a slot does not exceed the capacity of the slot, and each job i is assigned to at most l_i contiguous slots. Mathematically,* **MIA** *is given by the following integer program (IP).*

$$\begin{aligned} &\underset{A}{\operatorname{argmax}} \sum_{i \in N} \sum_{j \in M} v_{ij} \; \mathfrak{a}_{ij} \\ &s.t. \sum_{i \in N} \sum_{\substack{p \in M \\ j \in [p, p+l_i-1]}} \mathfrak{a}_{ip} \leqslant k, \forall j \in M, \\ &\sum_{j \in M} \mathfrak{a}_{ij} \leqslant 1, \forall i \in N; \; \mathfrak{a}_{ij} \in \{0,1\}, \forall i \in N, \forall j \in M \end{aligned} \tag{5}$$

The first set of inequalities insure that the number of job to be processed in a slot does not exceed the slot capacity k. We sum over every job $i \in N$ and check if it is under execution at j, for every $j \in M$. A job i is under execution at slot j, if it is allocated at a slot p s.t. $j \leqslant p + l_i - 1$. The second set of inequalities ensure that no job is allocated more than once.

We show that **MIA** is computationally intractable by performing a polynomial reduction from the Multi-Unit Combinatorial Auction (**MUCA**), which is NP-complete.

Description of **MUCA**: Consider a multiset $\mathcal{M} = (\mathcal{G}, y)$, where $\mathcal{G} = \{1, 2, 3, \ldots, g\}$ is a set of goods and y is a function, $y : \mathcal{G} \to \mathbb{Z}_{\geqslant 0}$ representing the multiplicity or the number of available units of the elements of $\mathcal{G}$ in $\mathcal{M}$. Each agent $i \in \mathcal{N} = \{1, 2, \ldots, n\}$ is a multi-minded bidder, which means i has a positive valuation $w_i(\cdot)$ for multiple bundles of available goods. We call the set of bundles for which agent i has a positive valuation to be the *demand set* of i, represented by $\mathcal{D}_i$. The valuation function is such that, $w_i(q) \in \mathbb{R}_{\geqslant 0}, \forall q \in \mathcal{D}_i$. We use the following notation $\mathcal{D} = [\mathcal{D}_i]_{i \in \mathcal{N}}$ and, $W_i = (w_i(q))_{q \in \mathcal{D}_i}$, $W = [W_i]_{i \in \mathcal{N}}$. In this paper we assume that, every agent demands at most one unit of every good. With this assumption, an allocation of a bundle of goods to the agents is represented as a matrix $\mathcal{S} = [\mathfrak{s}_{iq}]$, where $\mathfrak{s}_{iq} = 1$, if the bundle $q \in \mathcal{D}_i$ is allocated to i, else $\mathfrak{s}_{iq} = 0$. For an allocation $\mathcal{S}$, every agent i gets a valuation, $w_i(\mathcal{S})$=$\sum_{q \in \mathcal{D}_i} w_i(q) \; \mathfrak{s}_{iq}$, otherwise $w_i(\mathcal{S}) = 0$. The formal definition is as follows.

DEFINITION **6** (Multi-Unit Combinatorial Auction (**MUCA**)). *Given $(\mathcal{N}, \mathcal{M}, W, D)$, find an allocation $\mathcal{S}$ of goods to the agents such that $\sum_{i \in \mathcal{N}} \sum_{q \in \mathcal{D}_i} w_i(q) \; \mathfrak{s}_{iq}$ is maximum, and the total units of good $j \in \mathcal{G}$ allocated to the agents does not* *exceed j's availability $y(j)$, and every agent i is assigned at most one of the demanded bundle from $\mathcal{D}_i$. Mathematically,* **MUCA** *is given by the following integer program (IP):*

$$\begin{aligned} &\underset{\mathcal{S}}{\operatorname{argmax}} \sum_{i \in \mathcal{N}} \sum_{q \in \mathcal{D}_i} w_i(q)\, \mathfrak{s}_{iq} \\ &\textit{s.t.} \sum_{i \in \mathcal{N}} \sum_{\substack{q \in \mathcal{D}_i \\ j \in q}} \mathfrak{s}_{iq} \leqslant y(j), \forall j \in \mathcal{G} \qquad (6) \\ &\sum_{q \in \mathcal{D}_i} \mathfrak{s}_{iq} \leqslant 1, \forall i \in \mathcal{N};\ \mathfrak{s}_{iq} \in \{0,1\}, \forall i \in \mathcal{N}, \forall q \in \mathcal{D}_i \end{aligned}$$

The reduction of **MIA** to **MUCA** proceeds as follows. For a given instance (N, M, V, L, k) of **MIA**, construct an instance of **MUCA**$(\mathcal{N}, \mathcal{M}, W, \mathcal{D})$ problem such that, the set of agents $\mathcal{N}$ is N, the set of goods $\mathcal{G}$ is the set of the slots M within the period, where $y(j) = k, \forall j \in M$. For every $i \in \mathcal{N}$, the demand set $\mathcal{D}_i$ consists of $(m - l_i + 1)$ distinct bundles. Each of the bundles in $\mathcal{D}_i$ is of size l_i and consists of l_i contiguous slots. We denote a bundle as q_j if it contains l_i contiguous slots starting from slot j, and $w_i(q_j)$ is equal to v_{ij} (the valuation $i \in N$ gets if her job starts at slot $j \in M$). The above construction is done in polynomial steps of the input size. We construct a solution of **MIA** from a solution of **MUCA** in the following way: for every $q_j \in \mathcal{D}_i$ and $i \in \mathcal{N}$, if $\mathfrak{s}_{iq_j} = 1$ then, $\mathfrak{a}_{ij} = 1$ for every $i \in N$ and $j \in M$. Similarly, we construct a solution of **MUCA** from a solution **MIA** in the following way: if $\mathfrak{a}_{ij} = 1$ for $i \in N$ and $j \in M$ then, $\mathfrak{s}_{iq_j} = 1$ for every $q_j \in \mathcal{D}_i$ and $i \in \mathcal{N}$. The following lemma shows that an optimal solution of **MIA** is an optimal solution of **MUCA** and vice-versa.

LEMMA **1.** *Let $\mathcal{S}^*$ is a solution for* **MUCA** *for a multiset of goods $\mathcal{M}$, and $\mathbf{A}^*$ is such that, $\mathfrak{a}^*_{ij} = 1$ for $i \in N$ and $j \in M$, if and only if $\mathfrak{s}^*_{iq_j} = 1$ in $\mathcal{S}^*$ for $i \in \mathcal{N}$ and $q_j \in \mathcal{D}_i$, then A^* is an optimal solution for* **MIA** *if and only if $\mathcal{S}^*$ is an optimal solution for* **MUCA***.*

Since **MUCA** is NP-complete (Cramton et al. 2004; Rothkopf, Pekeč, and Harstad 1998), using Lemma 1, we get the following theorem.

THEOREM **1.** **MIA** *is NP-complete.*

However, it is possible to find an approximately efficient allocation in polynomial time that is truthful and individually rational. To find that, we leverage the approximation algorithm of **MUCA** due to Bartal et al. (2003, Theorem 3). Using Lemma 1 and the next few results, we prove that there exists a polynomial time truthful mechanism (**MIA** **A**pproximation **A**lgorithm or **MAA**) to achieve $O\big(km^{\frac{1}{k-2}}\big)$ approximation to the optimal solution of **MIA**.

The operational principle of **MAA** is a sequential dictatorship, where the sequence is an arbitrary order (WLOG $1, 2, \ldots, n$) of the agents and is independent of the information submitted by them. The mechanism comes with a price[6] vector which is updated while iterating over the agents in the sequence. We use a superscript i to denote the the price

[6]The terms *price* and *delay* are equivalent in the rest of the paper.

Algorithm 2: **MAA** in every period

```
1:  Procedure MAA($N, M, V, L, k$)
2:  $b \leftarrow \arg_i \max_{i \in N, j \in M} v_{ij}$; $v_{\max} \leftarrow \max_{i \in N, j \in M} v_{ij}$; //$b$ is the
    agent with highest valuation ($v_{max}$) for any slot.//
    $r \leftarrow (6m(k-1))^{\frac{1}{k-2}}$
3:  $\mathcal{Q}^0 \leftarrow [0, 0, \ldots, m$ times$]$
4:  $\mathfrak{a}^*_{bs'} = 1$, where $s' \leftarrow \operatorname{argmax}_{j \in M}(v_{bj})$; // For $b$ allocate
    its highest valued slot.//
    $\mathfrak{a}^*_{bj} = 0, \forall j \in M \setminus \{s'\}$; $\mathcal{P}_b = v^{-b}_{\max}$
5:  for $i = \{1, 2, \ldots, n\}$ and $i \neq b$ do
6:      for $j = \{1, 2, \ldots, m\}$ do
7:          $P^i_j \leftarrow \pi_0 \cdot r^{\mathcal{Q}^{i-1}_j}$
8:      end for
9:      $\mathfrak{a}^*_{is} = 1$, where $s \leftarrow \mathtt{max}(P^i, v_i)$ (Equation (7))
10:     $\mathfrak{a}^*_{ij} = 0, \forall j \in M \setminus \{s\}$
11:     $\mathcal{P}_i \leftarrow \sum_{j=s}^{s+l_i-1} P^i_j$
12:     for $j = \{1, 2, \ldots, m\}$ do
13:         if $j \in [s, s + l_i - 1]$ then
14:             $\mathcal{Q}^i_j \leftarrow \mathcal{Q}^{i-1}_j + 1$
15:         else
16:             $\mathcal{Q}^i_j \leftarrow \mathcal{Q}^{i-1}_j$
17:         end if
18:     end for
19: end for
20: return $\mathfrak{a}^*, \mathcal{P}$
```

faced by the agent i for slot j, P^i_j, when i's turn comes. Hence, $P^i = [P^i_j]_{j \in M}$ denotes the price vector seen by i. The mechanism also uses a function **max** that returns the slot s that maximizes agent i's utility given her valuation vector v_i and the price vector P^i when her job of length l_i starts from slot s. Mathematically, **max** is defined as follows:

$$\mathtt{max}(v_i, P^i) = \underset{s \in M}{\operatorname{argmax}}\, (v_{is} - \sum_{j \in [s, s+l_i-1]} P^i_j) \qquad (7)$$

MAA maintains a vector $\mathcal{Q}^i = [\mathcal{Q}^i_j]_{j \in M}$, where $\mathcal{Q}^i_j$ denotes the current allocated population of the slot j after allocating slots to i. First, **MAA** picks the agent b that has the maximum valuation $v_{\max}$ for any slot, and initializes $\mathcal{Q}^0_j = 0, \forall j \in M$. The initial price of every slot is set to $\pi_0 := \frac{v_{\max}}{6m(k-1)}$ and a constant factor $r = (6m(k-1))^{\frac{1}{k-2}}$ is defined. Consider an arbitrary order (WLOG $(1, 2, \ldots, n)$) of the agents. For each agent $i \neq b$ in sequence, the price P^i_j is computed using $\mathcal{Q}^{i-1}_j$, r, and π_0 such that the prices of the slots increase by a multiplicative factor of a suitable exponent of r such that the prices for more congested slots are higher. Then the highest utility-deriving slot s to start i's job is found using **max**, and the corresponding allocation vector for i is represented as $\mathfrak{a}^*_i$, where $\mathfrak{a}^*_{is} = 1, \mathfrak{a}^*_{ij} = 0, \forall j \neq s$. The total price (or delay) charged to i is denoted by $\mathcal{P}_i$ and is equal to $\sum_{j \in [s, s+l_i-1]} P^i_j$. The vector $\mathcal{Q}^i$ is updated after the allocation of slots to agent i. The agent b gets her most valued starting slot and pays the maximum valuation among all other agents and slots, which is represented by $v_{\max}^{-b}$.

An important feature of Algorithm 2 is that it *does not* explicitly check the capacity constraint. However, we show that the choices of π_0 and r implicitly maintains that in the following result. The units of slot j after Algorithm 2 executes that are occupied by agents except b are $\mathcal{Q}_j^*$.

LEMMA 2. *Let $\pi_0, r, \delta > 0$ be such that $\pi_0 r^\delta \geqslant v_{\max}$, then $\mathcal{Q}_j^* \leqslant \delta + 1$. This implies that* **MAA** *maintains the capacity constraints for each slot for $\delta = k - 2$.*

Proof. Assume for contradiction that $\mathcal{Q}_j^* > \delta + 1$ and let i be the first customer due to which this contradiction takes place for some slot j, i.e., $\mathcal{Q}_j^i > \delta + 1$. Since each customer does not get more than one unit of any slot, then it must be that $\mathcal{Q}_j^{i-1} > \delta$. Hence, for slot j, the following holds: $P_j^i > \pi_0 r^\delta \geqslant v_{\max} \geqslant \max_{j \in M} v_{ij}$. This makes i's total price for l_i contiguous slots including slot j to be more than her corresponding total valuation for those slots. This contradicts the definition of **max** since the utility becomes negative for agent i. □

The allocation to b is at most *one* unit from each slot. With carefully choosing π_0 and r, we bound the units of any slot allocated to all the other agents, $\mathcal{Q}_j^*$ to $(k-1)$, maintaining the possibility of the maximum use of every slot.

Since **max** allocates a slot only if that allocation increases the agent's utility, the following result holds.

THEOREM 2. **MAA** *is individually rational.*

Next, we show that misreporting the private information (v_i, l_i) is never beneficial for any agent i.

THEOREM 3. *In* **MAA***, reporting v_i and l_i truthfully in every period is a dominant strategy for all $i \in N$.*

Proof. For the agent b, we see that the utility is that of a second price auction and is independent of its length report. Since for second price auction revealing valuation truthfully is a dominant strategy therefore truthfully revealing valuation and length is a dominant strategy for b.

For the other agents, note that **MAA** considers the agents sequentially and allocates the utility-maximizing available slots in their turn. The order of the agents in **MAA** is independent of the valuations and lengths of the jobs. Consider agent i. When her turn comes, the mechanism picks the slots that give the maximum difference between the valuation of i for those slots and the current prices of those slots. Note that, the prices of those allocated slots are not dependent on the valuation or length reported by agent i (rather it is dependent on the reports of the previous agents in the sequence and b), and the mechanism allocates her the optimal set of slots. Hence, by misreporting the valuation v_i, agent i can either continue to get the same slots or get a worse set of slots w.r.t. her true valuation. Hence, there is no incentive for i to misreport her valuation.

Misreporting length: If $\hat{l}_i < l_i$, **MAA** allocates only $\hat{l}_i$ number of contiguous slots to i (which can have zero value as $\hat{l}_i$ is not sufficient for completion of her job) and i can get a negative payoff as she has to pay $\mathcal{P}_i$ (which is non-negative). If $\hat{l}_i > l_i$, then **MAA** allocates more slots than i actually needs. This allocation does not increase agent i's valuation, but increases the price since now she will be charged for $\hat{l}_i$ slots which is larger than the true length.

Combining the above two arguments that hold for all $i \in N$ irrespective of the reports of the other agents, we get the claim. □

To find the best slot for an agent, **max** checks the feasibility constraints and computes the allocation considering the valuation and the current price of the slots. As there are $(m - l_i + 1)$ possible allocations, **max** requires at most $O(m)$ time for every agent i. Therefore, the following result on the complexity of **MAA** holds.

THEOREM 4. **MAA** *has time complexity $O(mn)$.*

To show the approximation factor of **MAA**, we need a few more results. Due to paucity of space, we present those results (similar to the ones by Bartal et al. (2003), modified according to **MIA**) in supplementary material. These results help us prove the main theorem of this section.

THEOREM 5. *There exists a polynomial time $(O(mn))$, incentive compatible, and individually rational mechanism to achieve $O(km^{\frac{1}{k-2}})$ approximation to optimal solution for* **MIA***.*

The result above shows the existence of an approximately efficient mechanism that satisfies the other three desirable properties. The question of finding a lower bound on the approximation ratio remains open.

Experiments

While the mechanisms presented in this paper satisfy several desirable properties of a social scheduling mechanism for indivisible single and multiple slot jobs, its prioritizing profile for different classes of importance, costs of prioritization, and reduction in social congestion are not theoretically captured. In this section, we investigate these properties using real and synthetic datasets. The real dataset we collected from a store gave us only the checkout times. In absence of the length information of the visits, we resorted to the single-slot job model (with hourly slots) and tested the performance of **VCG-T** on this data (§). For multi-slot jobs, we simulated **MAA** to find the suboptimality and the reduction in the running time (§). For these experiments, we consider three discrete levels of valuations of the agents denoted by 3, 2, and 1, which can be interpreted as **high**, **medium**, and **low** respectively. The numbers represent the agent's valuation if they are allocated their most preferred slot. We used Gurobi (Gurobi Optimization 2020) under an academic license for all experiments.

Reduction in the social congestion

We consider a real data of customer footfall in a general store that we have collected from the store (the dataset will be made publicly available post publication). The dataset contains the customers' hourly checkout (billing) time from 7 AM to 9 PM (opening hours of the store) for the whole month of July 2020. Since the dataset was anonymized for customer identification, we have assumed that the billing timestamps are unique users for a day. Given the size of the store, around 32 people an hour should be a fair capacity to maintain social distance. However, the data shows that the monthly average during the periods 5-6 PM, 6-7 PM, and 7-8 PM were 38.00, 48.63, and 52.83 respectively. Interestingly, the monthly average of the population in an hour is 26.5, which is well within the safety limits. Therefore, this dataset works as a perfect example where users can benefit significantly from social scheduling.

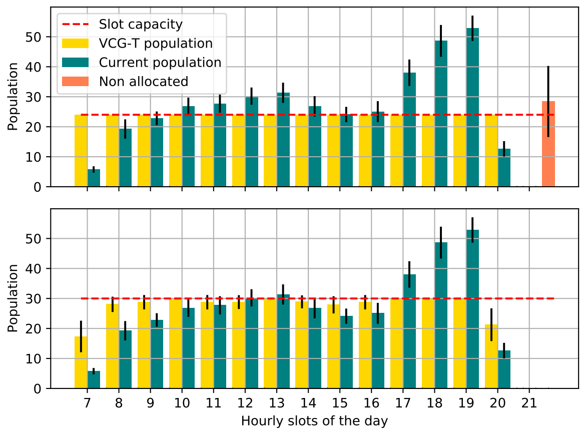

Figure 1: Social congestion reduction, slot capacities 24 (top) and 30 (bottom).

We divide the store opening hours into 14 hourly slots between 7 AM to 9 PM. For each day the slots are sorted in decreasing order of footfall on that day. We fix this order of the slots as the preference order of each agent for that day. The valuations of every customer for her most preferred slot is drawn from a distribution $\{$**high**:0.1, **medium**:0.3, **low**:0.6$\}$. The valuations for the other slots are assumed to decrease with a multiplicative factor $\delta = 0.5$ in the order of the slot preference, i.e., the valuation for the t-th most preferred slot of a **medium** agent will be $2\delta^{t-1}$. In this experiment, if a user is not allocated a slot on a certain day, she is given an option to update her importance and preferences for the next day. For the experiments, we assume that the user increases the importance by one level, e.g., **low** becomes **medium**, and keeps the slot preferences the same. After three consecutive days, if an agent is not allocated, she is considered 'non-allocated,' and alternative arrangements (e.g., home delivery) are made. Figure 1 shows the comparison of the average current population with that allocated by **VCG-T** for slot capacities of 24 (above) and 30 (below). The figures also show the daily non-allocated population in red. Each plot in this section shows the average values with 95% confidence interval. The plots show the trade-off between better social distancing (lower slot capacity) and its cost (non-allocation). However, in both these cases, the social congestion is reduced by approximately 50% during the rush hours.

VCG-T also prioritizes the jobs at a cost. Figure 2 shows the allocated slot preference and the delays for the three difference classes of valuations for the slot capacity 30. It shows that a higher valuation comes with a higher delay.

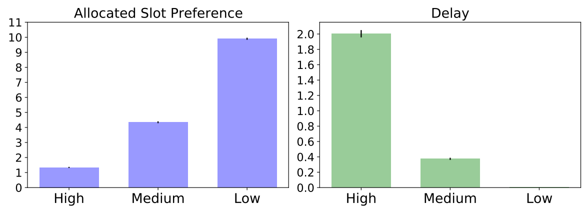

Figure 2: Priority and delay trade-off of **VCG-T**.

Suboptimality vs Complexity Reduction (MAA)

The sub-optimality of **MAA** (Algorithm 2) was obtained for a worst-case scenario in §. Here we investigate the sub-optimality of **MAA** and the amount of time it reduces w.r.t. a brute-force algorithm that finds the optimal allocation of the slots. The top plot of Figure 3 shows the percentage reduction $((t_{\texttt{OPT}} - t_{\texttt{MAA}})/t_{\texttt{OPT}})$ in the running time of **MAA** compared to the optimal mechanism, where $t_{\texttt{OPT}}$ and $t_{\texttt{MAA}}$ are the running times of the optimal and **MAA** mechanisms respectively. The bottom plot shows the ratio of the optimal social welfare to the welfare yielded by **MAA**. For each agent a slot preference order is generated uniformly at random from the set of all feasible preference orders over the slots. Valuations for most preferred slot is generated from a uniform distribution over $\{$**high**, **medium**, **low**$\}$ and for other slots is assumed to decrease with a multiplicative factor $\delta = 0.5$, same as that in §. The length of jobs are generated randomly with uniform distribution in range 1 to m. The experiment is run with $n = 6, k = 5$, and m varying from 3 to 8. For each value of m, the experiment is repeated for 100 valuation matrix and length vector pairs. The parameters m, n, k are chosen such that the optimal mechanism is computable in a reasonable time, yet the experiment yields an insightful result. We see that **MAA** reduces the running time by more than 99.5% and yields an approximation of roughly 1.75 on an average.

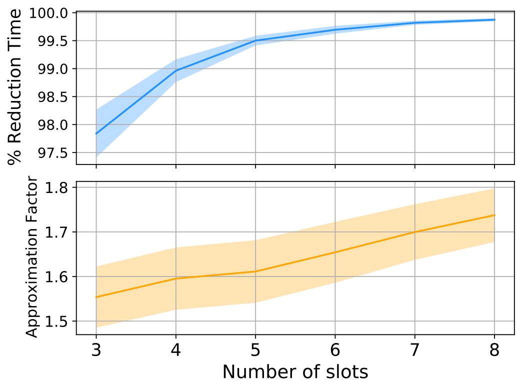

Figure 3: Running time and approximation factor trade-off for **MAA**.

Acknowledgments

Nath is supported by a MATRICS grant (MTR/2021/000367) from Science and Engineering Research Board, Govt. of India, a Technocraft Center for Applied Artificial Intelligence fund (DO/2021-TCAI002-009), an Indian Institute of Technology Bombay fund (2022058).

References

Bartal, Y.; Gonen, R.; and Nisan, N. 2003. Incentive Compatible Multi Unit Combinatorial Auctions. In *Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '03, 72–87. New York, NY, USA: Association for Computing Machinery. ISBN 1581137311.

Bergemann, D.; and Välimäki, J. 2010. The Dynamic Pivot Mechanism. *Econometrica*, 78: 771–789.

Bloch, F. 2017. Second-best mechanisms in queuing problems without transfers: The role of random priorities. *Mathematical Social Sciences*, 90: 73–79.

Börgers, T. 2015. *An introduction to the theory of mechanism design*. Oxford University Press, USA.

Braverman, M.; Chen, J.; and Kannan, S. 2016. Optimal provision-after-wait in healthcare. *Mathematics of Operations Research*, 41(1): 352–376.

Camion, P. 1965. Characterization of totally unimodular matrices. *Proceedings of the American Mathematical Society*, 16(5): 1068–1073.

Cavallo, R. 2021. Social Distancing Equilibrium. Unpublished.

Chen, X.; Hu, X.; Liu, T.-Y.; Ma, W.; Qin, T.; Tang, P.; Wang, C.; and Zheng, B. 2016. Efficient mechanism design for online scheduling. *Journal of Artificial Intelligence Research*, 56: 429–461.

Clarke, E. H. 1971. Multipart Pricing of Public Goods. *Public Choice*, 11: 17–33.

Cramton, P.; Shoham, Y.; Steinberg, R.; et al. 2004. Combinatorial auctions. Technical report, University of Maryland, Department of Economics-Peter Cramton.

Fong, M. W.; Gao, H.; Wong, J. Y.; Xiao, J.; Shiu, E. Y.; Ryu, S.; and Cowling, B. J. 2020. Nonpharmaceutical measures for pandemic influenza in nonhealthcare settings—social distancing measures. *Emerging infectious diseases*, 26(5): 976.

Ghosh, S.; Long, Y.; and Mitra, M. 2020. Prior-free online mechanisms for queueing with arrivals. *Economic Theory*, 1–30.

Glass, R. J.; Glass, L. M.; Beyeler, W. E.; and Min, H. J. 2006. Targeted social distancing designs for pandemic influenza. *Emerging infectious diseases*, 12(11): 1671.

Grötschel, M.; Lovász, L.; and Schrijver, A. 1993. Complexity, oracles, and numerical computation. In *Geometric Algorithms and Combinatorial Optimization*, 21–45. Springer.

Groves, T. 1973. Incentives in Teams. *Econometrica*, 41: 617–631.

Gurobi Optimization, L. 2020. Gurobi Optimizer Reference Manual.

Hajiaghayi, M. T. 2005. Online Auctions with Re-Usable Goods. In *Proceedings of the 6th ACM Conference on Electronic Commerce*, EC '05, 165–174. New York, NY, USA: Association for Computing Machinery. ISBN 1595930493.

Hatchett, R. J.; Mecher, C. E.; and Lipsitch, M. 2007. Public health interventions and epidemic intensity during the 1918 influenza pandemic. *Proceedings of the National Academy of Sciences*, 104(18): 7582–7587.

Koutsoupias, E. 2014. Scheduling without payments. *Theory of Computing Systems*, 54(3): 375–387.

Langer, T.; and Weber, M. 2005. Myopic prospect theory vs. myopic loss aversion: how general is the phenomenon? *Journal of Economic Behavior & Organization*, 56(1): 25–38.

Lavi, R.; and Nisan, N. 2004. Competitive analysis of incentive compatible on-line auctions. *Theoretical Computer Science*, 310(1): 159–180.

Leclerc, F.; Schmitt, B. H.; and Dube, L. 1995. Waiting time and decision making: Is time like money? *Journal of Consumer Research*, 22(1): 110–119.

Mitra, M. 2001. Mechanism design in queueing problems. *Economic Theory*, 17(2): 277–305.

Roberts, K. 1979. The characterization of implementable choice rules. *Aggregation and revelation of preferences*, 12(2): 321–348.

Rothkopf, M. H.; Pekeč, A.; and Harstad, R. M. 1998. Computationally manageable combinational auctions. *Management science*, 44(8): 1131–1147.

Schrijver, A. 2003. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media.

Shoham, Y.; and Leyton-Brown, K. 2008. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

Thunström, L.; Newbold, S. C.; Finnoff, D.; Ashworth, M.; and Shogren, J. F. 2020. The benefits and costs of using social distancing to flatten the curve for COVID-19. *Journal of Benefit-Cost Analysis*, 1–27.

Toxvaerd, F. 2020. Equilibrium Social Distancing. Cambridge Working Papers in Economics 2021, Faculty of Economics, University of Cambridge.

Vickrey, W. 1961. Counter Speculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance*, 16(1): 8–37.

Wilder-Smith, A.; and Freedman, D. O. 2020. Isolation, quarantine, social distancing and community containment: pivotal role for old-style public health measures in the novel coronavirus (2019-nCoV) outbreak. *Journal of travel medicine*.

Appendix

VCG−T for Single-slot Jobs

We first show that the allocation given by **VCG−T** indeed maximizes per-period social welfare.

THEOREM **6.** *The allocation of* **VCG−T** *given by LP (3) always gives integral solutions.*

Proof. Consider the vector $x^\top = (a_{11}, \dots, a_{1m}, \dots, a_{n1}, \dots, a_{nm})$, i.e., the rows of A linearized as a vector. We can write the constraints of LP (3) in using a $(n+m) \times nm$ constraint matrix, s.t.,

$$\begin{pmatrix} 1 & \dots & 1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 1 & 0 & \dots & 0 \\ & & & & \dots & & & & \\ 1 & \dots & 0 & 1 & \dots & 0 & 1 & \dots & 0 \\ 0 & 1 & \dots & 0 & 1 & \dots & 0 & 1 & 0 \\ & & & & \dots & & & & \end{pmatrix} x \leqslant \begin{pmatrix} 1 \\ \vdots \\ k \\ \vdots \end{pmatrix}$$

Denote the matrix on the LHS by C. The first n and the next m rows correspond to the first and second set of constraints of LP (3) respectively. We show that C is totally unimodular (TU), which is sufficient to conclude that LP (3) has integral solutions. We use the Ghouila-Houri characterization (Camion 1965) to prove that C is TU. According to this characterization, a $p \times q$ matrix C is TU if and only if each set $R \subseteq \{1, 2, \cdots, p\}$ can be partitioned into two sets R_1 and R_2, such that, $\sum\limits_{i \in R_1} a_{ij} - \sum\limits_{i \in R_2} a_{ij} \in \{1, 0, -1\}$, for $j = 1, 2, \cdots, q$. Note that, in C every column has two 1's, one in the first n rows and one in the next m rows. Pick any subset R of the rows, construct the R_1 to be the rows that come from the first n rows, and R_2 to be the rows that come from the last m rows (one of these partitions can be empty). Clearly, the difference in each column of the rows R will lie in $\{1, 0, -1\}$. Hence proved. □

The result above shows that the optimal solution of LP (3) is an optimal solution of the corresponding integer program that maximizes the per-period social welfare. Hence, we conclude the following.

COROLLARY **1.** **VCG−T** *is EPP.*

Even though the LP formulation of **VCG−T** is without loss of optimality, in general, LPs can be weakly polynomial, i.e., the space used by the algorithm may not be bounded by a polynomial in the size of the input. However, we show that an even stronger result holds for **VCG−T**. The forthcoming results show that the allocation and delays of **VCG−T** are strongly polynomial. To show this, we will first reduce the allocation problem (LP (3)) to a minimum weight b-matching problem, which is known to be strongly polynomial (Schrijver 2003).

Consider an edge-weighted bipartite graph (N, M, E), where N and M are the agent set and set of slots respectively. The set E denotes the edges (i, j) with weights $-v_{ij}$. The matching constraints are given by $b_i = 1, \forall i \in N$, and $b_j = k, \forall j \in M$.

LEMMA **3.** *Let $E^* \subseteq E$ be a perfect b-matching in (N, M, E) and $A^* = [a^*_{ij}]_{i \in N, j \in M}$ be an allocation where $a^*_{ij} = 1 \Leftrightarrow (i, j) \in E^*$. The matching E^* is a minimum weight perfect b-matching iff A^* is an optimal solution to LP (3).*

Proof. We prove this via contradiction. Suppose A^* is not an optimal solution to LP (3), i.e., there exists A' which satisfies the constraints and yet gives a larger value to the objective function than that of A. Hence, $\sum_{j \in M} \sum_{i \in N} v_{ij} a'_{ij} > \sum_{j \in M} \sum_{i \in N} v_{ij} a^*_{ij}$. Consider the edge set E' corresponding to A'. Clearly this is a perfect b-matching, since A' satisfies the constraints of LP (3), and E' gives a lower weight than E^*, which proves that E^* is not the minimum weight perfect b-matching. The implications can be reversed to obtain the other direction of the proof. □

Note that the delays are calculated by solving an equivalent LP like LP (3) with one less agent. Therefore, each of these LPs is strongly polynomial, and the planner needs to solve n of them. The computation of each delay needs the addition of $2(n-1)$ terms and one subtraction. Hence, the number of computations is polynomial in the number of numbers in the input instance, and the space required is polynomial in the input size. Therefore we conclude the following.

COROLLARY **2.** *The computation of the delays in* **VCG−T** *is strongly polynomial.*

Combining Theorem 6, Lemma 3, and Corollary 2, we get the following result.

THEOREM **7.** **VCG−T** *provides a combinatorial, strongly polynomial algorithm for computing a social schedule and delays.*

Since **VCG−T** uses the VCG payment expression to compute the time delay and because the allocated slots are *goods* to the agents, the following two facts follow from the known properties of the VCG mechanism.

FACT **1.** **VCG−T** *is per period dominant strategy truthful.*

Proof. This proof is a standard exercise in the line of the proof for Vickery-Clarke-Groves (VCG) mechanism (Vickrey 1961; Clarke 1971; Groves 1973).

Let us assume for the contradiction that, there exist an agent i for having true valuations for the slots as, v_i, but misreports it as v'_i(the corresponding value function is v'_i), and gets better utility. Suppose $A(v'_i, v_{-i}) = A'$ and $A(v_i, v_{-i}) = A^*$. The utility of i for A' is:

$$\begin{aligned} & v_i(A') - d_i(v'_i, v_{-i}) \\ & = v_i(A') - \sum_{\ell \in N \setminus \{i\}} v_\ell(A(v_{-i})) + \sum_{\ell \in N \setminus \{i\}} v_\ell(A') \\ & = \sum_{\ell \in N} v_\ell(A') - \sum_{\ell \in N \setminus \{i\}} v_\ell(A(v_{-i})) \end{aligned}$$

Similarly, the utility of i for A^* is:

$$= \sum_{\ell \in N} v_\ell(A^*) - \sum_{\ell \in N \setminus \{i\}} v_\ell(A(v_{-i}))$$

If i gets better utility by misreporting her valuation as $v'(.)$, then

$$\sum_{\ell \in N} v_\ell(A') > \sum_{\ell \in N} v_\ell(A^*)$$

The above inequality leads to the contradiction that A^* is optimal for the reported valuation (v_i, v_{-i}). Therefore, **VCG-T** is dominant strategy truthful in every period. □

FACT **2.** **VCG-T** *is individually rational for every agent.*

Proof. This proof is a standard exercise in the line of the proof for Vickery-Clarke-Groves (VCG) mechanism (Vickrey 1961; Clarke 1971; Groves 1973).

Let us assume for the contradiction that, there exist an agent i for having true valuations for the slots as, v_i, but misreports it as v'_i(the corresponding value function is v'_i), and gets better utility. Suppose $A(v'_i, v_{-i}) = A'$ and $A(v_i, v_{-i}) = A^*$. The utility of i for A' is:

$$\begin{aligned} & v_i(A') - d_i(v'_i, v_{-i}) \\ & = v_i(A') - \sum_{\ell \in N \setminus \{i\}} v_\ell(A(v_{-i})) + \sum_{\ell \in N \setminus \{i\}} v_\ell(A') \\ & = \sum_{\ell \in N} v_\ell(A') - \sum_{\ell \in N \setminus \{i\}} v_\ell(A(v_{-i})) \end{aligned}$$

Similarly, the utility of i for A^* is:

$$= \sum_{\ell \in N} v_\ell(A^*) - \sum_{\ell \in N \setminus \{i\}} v_\ell(A(v_{-i}))$$

If i gets better utility by misreporting her valuation as $v'(.)$, then

$$\sum_{\ell \in N} v_\ell(A') > \sum_{\ell \in N} v_\ell(A^*)$$

The above inequality leads to the contradiction that A^* is optimal for the reported valuation (v_i, v_{-i}). Therefore, **VCG-T** is dominant strategy truthful in every period. □

Proof of Lemma 1

Proof. Suppose the above statement is not true and hence $\mathbf{A}'$ but not $\mathbf{A}^*$ is an optimal solution for **MIA**.

$$\sum_{i \in N} \sum_{j \in M} v_{ij}\ \mathfrak{a}'_{ij} \geqslant \sum_{i \in N} \sum_{j \in M} v_{ij}\ \mathfrak{a}^*_{ij}$$

As $w_i(q_j) = v_{ij}$, and $\mathfrak{a}^*_{ij} = 1$ only if $\mathfrak{s}^*_{iq_j} = 1$, with the constructed $\mathcal{S}'$ corresponding to $\mathbf{A}'$ the following inequality holds,

$$\sum_{i \in \mathcal{N}} \sum_{q_j \in \mathcal{D}_i} w_i(q_j)\ \mathfrak{s}'_{iq_j} \geqslant \sum_{i \in \mathcal{N}} \sum_{q_j \in \mathcal{D}_i} w_i(q_j)\ \mathfrak{s}^*_{iq_j}$$

The above equation results in a contradiction that $\mathcal{S}^*$ is an optimal solution for **MUCA**.

Since each step of the above proof has implications in both directions, the other direction of the proof is implied. □

Restated results from the literature

In this section, we will restate a few results from (Bartal, Gonen, and Nisan 2003), which will help us prove the approximation factor in Theorem 5. The lemma and section numbers of these results in the original paper are mentioned within parentheses in the restated lemmata.

LEMMA **4** ((Bartal, Gonen, and Nisan 2003, Section 4.2, Lemma 4))**.** *For every agent i, $v_i(\mathfrak{a}^*_i) \geqslant v_i(\mathfrak{a}'_i) - \sum_{j \in [s, s+l_i-1]\ s.t.\ \mathfrak{a}'_{is}=1} P^*_j$ for every allocation $\mathfrak{a}'_i$, where, P^* is the vector of prices of slots at the end of Algorithm 2.*[7]

Let $V(ALG)$ and $V(OPT)$ denotes the sum of valuations of customers for the allocation $\mathbf{A}^*$ given by **MAA**, and that for the optimal allocation (say $\hat{\mathbf{A}}$) respectively. Similarly, $V(ALG^{-b})$ and $V(OPT^{-b})$ represents the sum of valuations of every agent except b according to $\mathbf{A}^*$ and $\hat{\mathbf{A}}$ respectively. Summing it for all the agent $i \in N$, we get the following corollary.

COROLLARY **3.** $V(ALG^{-b}) \geqslant V(OPT^{-b}) - (k-1) \sum_{j \in M} P^*_j$

The following result provides a lower bound on $V(ALG^{-b})$.

LEMMA **5** ((Bartal, Gonen, and Nisan 2003, Section 4.2, Lemma 5))**.** $V(ALG^{-b}) \geqslant \frac{\sum_{j \in M} P^*_j - m\pi_0}{r-1}$

Combining Lemma 5 and Corollary 3, we state the following result.

LEMMA **6.** *If* $m(k-1)\pi_0 \geqslant \frac{V(OPT^{-b})}{2}$*, then* $2((k-1)(r-1)+1) \geqslant V(OPT^{-b})/V(ALG^{-b})$.

Following the conditions in Lemma 2 and Lemma 6, we fix $\pi_0 = \frac{v_{\max}}{6m(k-1)}$, $r = (6m(k-1))^{\frac{1}{k-2}}$. We restate the following result about the approximation ratio from (Bartal, Gonen, and Nisan 2003).

LEMMA **7** ((Bartal, Gonen, and Nisan 2003, Section 5, Lemma 8))**.** *The approximation ratio of Algorithm 2 is* $3((k-1)(r-1)+1)$.

Finally, combining Lemmas 6 and 7, we get Theorem 5.

Additional Experimental Results

Prioritizing profile and its cost

In this section, we investigate what the typical priority slots allotted to an agent of a specific class in **VCG-T** are. The top plot of Figure 4 shows the agents' allocated slot preferences (mean with one standard deviation) versus the population (n) plot where $m = 5, k = 4$, and $\delta = 0.65$. The importance of an agent is picked uniformly at random. Values of n vary between 2 to $1.1mk$ in steps of 1 (for a population beyond mk, some agents have to be dropped). The experiment is repeated 100 times for every n. The plot shows that the higher the importance, the lower is the allocated slot preference for the agents, which is desirable.

[7]With a slight abuse of notation, we denote $[a, b]$ to be the integers between a and b, where $a < b$.

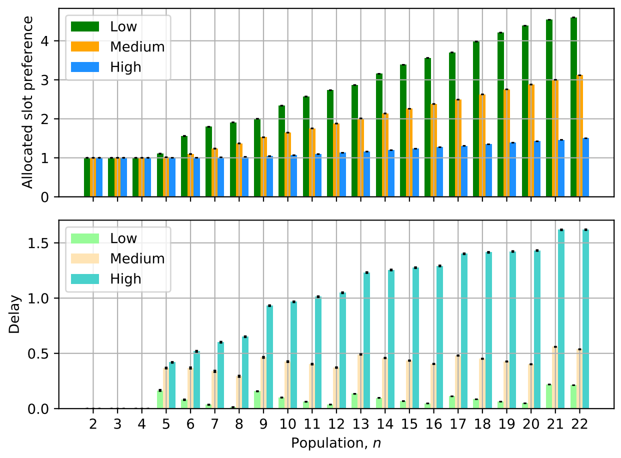

Figure 4: Priority-delay trade-off for **VCG–T**.

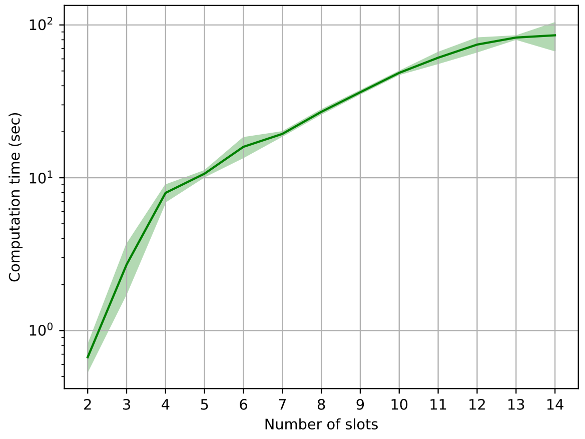

Figure 5: Computation times of **VCG–T**.

However, **VCG–T** does the prioritized allocations of the agents at the cost of their delays. The bottom plot of Figure 4 shows the corresponding delays decided by **VCG–T** for each of these three classes. The plot shows that an early slot allocation of an agent because of her importance also comes with a longer delay and shows the trade-off between these two decisions.

Scalability

This section examines **VCG–T**'s computation time for finding the allocation and delays for a realistic population. We run **VCG–T** in Python for different number of slots (m) with slot capacity (k) being 12. For every m, we fixed $n = mk$ and repeated the experiment $10n$ times. Figure 5 shows the growth of the computation time of the mechanism. As a reference, to solve the allocation and delays for the store to study reduction in social congestion (Figure 1), it takes about 100 secs. The simulations have been performed in a 64-bit Ubuntu 18.04 LTS machine with Intel(R) Core(TM) i7-7700HQ CPU @2.80GHz quad-core processors and 16 GB RAM.