Social Distancing via Social Scheduling

Deepesh Kumar Lall¹, Garima Shakya¹, and Swaprava Nath¹

¹Indian Institute of Technology Kanpur, {deepeshk, garimas, swaprava}@iitk.ac.in

October 25, 2021

Abstract

Motivated by the need of 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. Naturally, the formulation applies to several similar problems. We consider indivisible 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) aims to prioritize 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. We show that the problem becomes NP-complete for indivisible multi-slot demands, and 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.

1 Introduction

Pandemics like the COVID-19 showed that one of the most effective solutions against infectious diseases is social distancing [38]. Therefore, it is a practice we ought to master and stay prepared as a preventive disease containment measure in the future. While most citizens are willing to follow social distancing, the lack of communication and coordination among them, particularly prior to an immediate lockdown, overcrowds shopping centers even though the footfalls could have been evenly distributed over the shop’s working hours to maintain a medically recommended social distance.

In our setting, each customer has an infinite queue of jobs that have different importances (privately known only to the customer) and lengths. However, the customers are myopic, i.e., worries only about the last unprocessed job. 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 [2] or multi-ownership computational jobs in a single-core processor [31]. 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.

1.1 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 unprocessed job.\(^1\) 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., transfers of payoff), only *dictatorial* mechanisms (where a pre-selected agent’s favorite outcome is always selected) are truthful [28, 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 [24]. 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 [34, 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:

- Even though maximizing the *social welfare* (sum of the agents’ valuations) of the slot allocation is a combinatorial optimization problem; we show that this problem is computationally easy to solve for jobs with single-slot length (Theorems 1 and 2).
- We show that the standard Vickrey-Clarke-Groves (VCG) payment [12, 19, 37] can be used as the delay (cooling-off time) in this setting to ensure truthfulness (Fact 1) and participation of the agents (Fact 2). We call the allocation and delay together as the mechanism VCG–T (VCG with Time delays).
- The welfare maximizing allocation of multi-slot jobs are computationally hard (Theorem 3). We propose a polynomial time mechanism (Theorem 6) which ensures participation (Theorem 4), truthfulness (Theorem 5), and is approximately optimal (Theorem 7).
- Our real and synthetic data experiments (Section 8) 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 (Section 8.1). For the multi-slot jobs, our approximately optimal mechanism provides a reasonable approximation at a much reduced computational cost in practice (Section 8.2).

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

\(^1\)A typical shopper knows that she needs to visit a store many times but precisely knows the importance of the immediate visit.
1.2 Related Work

Social distancing measures have been widely successful and recommended for pandemic control [14, 16, 35]. However, it is also observed that the benefits of social distancing depend on the extent to which the citizens follow it. An extensive part of the recent research related to social distancing during COVID19 aims to understand the relationship of social distancing with different public policies and other factors [1, 11, 17, 26]. Individuals are sometimes reluctant to pay the costs inherent in social distancing which can limit its effectiveness as a control measure [27]. Toxvaerd [36] considers social distancing in the susceptible-infected-recovered (SIR) epidemiological model and shows that “during the equilibrium social distancing phase, individuals gradually reduce their social distancing efforts despite the infection probability not decreasing”. Cavallo [9] shows that under an uncoordinated model, every equilibrium involves more social contact than it occurs in a social optima. A related thread of social distancing measures exists in workforce-intensive organizations in the forms of rostering, workplace design, or allowing people to work from home [30]. However, these measures are centrally controlled, and individuals do not have a scope to behave strategically. On the other hand, the businesses maintain social distancing by enforcing certain appointment booking apps (web or smartphone-based) on customers. These apps run a first-come-first-served algorithm for slot booking and do not provide any priority preservation guarantees.²

Coordination among the citizens is essential for social distancing [33], but the technology is still not much developed to address the social coordination problem, particularly when the participants are self-interested independent decision makers. Mechanism design is an approach which can equip an artificially intelligent app to satisfy certain desirable properties on the face of the participants’ strategic behavior. This is attempted in the current paper.

Resource allocation with monetary transfers to ensure truthfulness, e.g., in the context of jobshop scheduling [21], has a rich literature that is close to our work. Lavi and Nisan [23] 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. [10] 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 [5, 15, 25], while our model can admit costs of any structure.

In the context of job scheduling without money, Koutsoupias [22] 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. [7] 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.

2 Single-slot Job Allocation Setup

Define $N := \{1, \ldots, n\}$ to be the set of agents that are trying to access a facility $F$. 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

²An example app is www.appointy.com.
slot has a maximum capacity of $k$, which is decided by the region’s social distancing norm based on the size of the facility. 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 $v_{ij} \in \mathbb{R}_{\geq 0}$ (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 $v_i = (v_{ij}, j \in M) \in \mathbb{R}^M_{\geq 0}$. In this paper, we consider different 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 [34], which implies that every agent wants a more valued slot to be assigned to her and also wants to wait less.

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

Denote the set of all allocations by $A$. The delays $d_i \in \mathbb{R}_{\geq 0}, \forall i \in N$. 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 $f : \mathbb{R}^{m \times n} \rightarrow A \times \mathbb{R}^n$ 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.$^4$

### 3 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

$$A^*(v) \in \arg \max_{A \in A} \sum_{i \in N} \sum_{j \in M} v_{ij} a_{ij}.$$ \hfill (2)

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

$^4$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 \( i \in N \)

\[
v_i(A(v_i, \hat{v}_{-i})) - d_i(v_i, \hat{v}_{-i}) \geq v_i(A(\hat{v}_i, \hat{v}_{-i})) - d_i(\hat{v}_i, \hat{v}_{-i}).
\]

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 \( i \in N \)

\[
v_i(A(v)) - d_i(v) \geq 0.
\]

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. There are problems for which a quick method of solving them is not known yet. Still, a given solution can be verified quickly in time polynomial of the input size, which means that the given solution’s validity can be tested quickly. The complexity class of such problems is known as NP (nondeterministic polynomial). We consider the NP-complete class of problems, which is defined as follows.

**Definition 5 (NP-complete)** A decision problem \( Q \) is NP-complete if:

1. \( Q \) is in NP (class of all decision problems verifiable in polynomial time), and
2. Every problem in NP is reducible to \( Q \) in polynomial time.

We will show that the slot allocation problem in a certain setting belongs to this class.

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 [18]. The arithmetic model of computation defines strongly polynomial algorithms. It is assumed that the basic arithmetic operations (addition, subtraction, multiplication, division, and comparison) take a unit time step to perform, regardless of the operands’ sizes.

**Definition 6 (Strongly Polynomial)** An algorithm runs in strongly polynomial time if [18]

\[
\triangleright \text{the number of operations in the arithmetic model of computation is bounded by a polynomial in the number of operands in the input instance; and}
\]

\[
\triangleright \text{the space used by the algorithm is bounded by a polynomial in the input size.}
\]

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 \( v_i \)'s or \( k \).
4 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. 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 [6] 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, our proposed 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 allocation as follows.

\[
\arg\max_A \sum_{j \in M} \sum_{i \in N} v_{ij}a_{ij} \\
\text{s.t. } \sum_{j \in M} a_{ij} \leq 1, \forall i \in N; \sum_{i \in N} a_{ij} \leq k, \forall j \in M
\]

(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 $A^*(\hat{v})$. 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,

\[
d_i := \sum_{\ell \in N \setminus \{i\}} v_{\ell}(A^*_{-i}) - \sum_{\ell \in N \setminus \{i\}} v_{\ell}(A^*)
\]

(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 $d(\hat{v})$

In the following few sections, we present the theoretical results related to single and multi-slot jobs.

5 Single-slot Jobs

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

---

5 For mechanisms that consider the dynamic extension of such allocation problems with finite or infinite horizon [4, 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.
Theorem 1  The allocation of VCG-T given by LP (3) always gives integral solutions.

Proof: Consider the vector \( x^\top = (a_{11}, \ldots, a_{1m}, a_{n1}, \ldots, a_{nm}) \), 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.,

\[
\begin{pmatrix}
1 & \ldots & 1 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
0 & \ldots & 0 & 1 & \ldots & 1 & 0 & \ldots & 0 \\
1 & \ldots & 0 & 1 & \ldots & 0 & 1 & \ldots & 0 \\
0 & 1 & \ldots & 0 & 1 & \ldots & 0 & 1 & 0 \\
\vdots & & & & & & & & \vdots
\end{pmatrix}
\begin{pmatrix}
x \leq \end{pmatrix}
\begin{pmatrix}
1 \\
\vdots \\
k \\
\vdots
\end{pmatrix}
\]

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 [8] 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, \( \sum_{i \in R_1} a_{ij} - \sum_{i \in R_2} a_{ij} \in \{1, 0, -1\} \), 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 [32].

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 1  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 1, Lemma 1, and Corollary 2, we get the following result.

**Theorem 2** 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 [12, 19, 37].

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:

\[
v_i(A') - d_i(v'_i, v_{-i})
= v_i(A') - \sum_{\ell \in N \setminus \{i\}} v_\ell(A(v_{-i})) + \sum_{\ell \in N \setminus \{i\}} v_\ell(A')
= \sum_{\ell \in N} v_\ell(A') - \sum_{\ell \in N \setminus \{i\}} v_\ell(A(v_{-i}))
\]

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 \( v'(\cdot) \), 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. \( \square \)

**Fact 2** VCG-T is individually rational for every agent.
Proof: Consider agent $i$. The utility of $i$ under VCG-T is $v_i(A^*(v)) - d_i(v)$

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

$$= \left( \sum_{\ell \in N} v_\ell(A^*) - \sum_{\ell \in N} v_\ell(A^*_{-i}) \right) + v_i(A^*_{-i}) \geq 0$$

The second equality holds by reorganizing the terms and adding and subtracting $v_i(A^*_{-i})$. Note that the difference term in the parentheses in the last expression is always non-negative since $A^*$ is the optimal allocation for all allocations. In particular, $A^*_{-i}$ is also a feasible allocation when agent $i$ is present. The term $v_i(A^*_{-i})$ is zero. Hence the inequality follows. ■

The multi-slot jobs, unlike the single-slot jobs, are relatively difficult to schedule, as we discuss in the following section.

6 Multi-slot Jobs

In this section, we consider jobs with different lengths, i.e., for agent $i$, the job may be of length $l_i \geq 1$. 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 $A = [a_{ij}]$, where $a_{ij} = 1$ if agent $i$’s job starts at slot $j$, else $a_{ij} = 0$, and $a_i$ represents the slot allocation vector for agent $i$. Keeping all other notations as before, we define the MIA problem as follows.

**Definition 7 (Multi-slot Indivisible jobs Allocation problem (MIA))**: Given $(N, M, V, L, k)$, find an allocation $A$, such that $\sum_{i \in N} \sum_{j \in M} v_{ij}(a_{ij})$ 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).

$$\arg\max_A \sum_{i \in N} \sum_{j \in M} v_{ij} a_{ij} \quad \text{s.t.} \quad \sum_{i \in N} \sum_{j \in M} a_{ip} \leq k, \forall j \in M,$$

$$\sum_{p \in M} a_{ip} \leq k, \forall j \in [p, p+l_i-1],$$

$$\sum_{j \in M} a_{ij} \leq 1, \forall i \in N; \quad a_{ij} \in \{0,1\}, \forall i \in N, \forall j \in M \quad (5)$$

9
In the first set of inequalities, 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. \( j \leq p + l_i - 1 \). The second set of inequalities ensure that no job is allocated more than once.

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

**Description of \textsc{MUCA}:** Consider a multiset \( \mathcal{M} = (\mathcal{G}, y) \), where \( \mathcal{G} = \{1, 2, 3, \ldots, q\} \) is a set of goods and \( y : \mathcal{G} \to \mathbb{Z}_{\geq 0} \) representing the multiplicity or the number of available units of the elements of \( \mathcal{G} \) in \( \mathcal{M} \). Each agent \( i \in N = \{1, 2, \ldots, n\} \) 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 \( D_i \). The valuation function is such that, \( w_i(q) \in \mathbb{R}_{\geq 0}, \forall q \in D_i \). We use the following notation \( D = [D_i]_{i \in N} \) and, \( W_i = (w_i(q))_{q \in D_i}, W = [W_i]_{i \in N} \).

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 \( S = [s_{iq}] \), where \( s_{iq} = 1 \), if the bundle \( q \in D_i \) is allocated to \( i \), else \( s_{iq} = 0 \). For an allocation \( S \), every agent \( i \) gets a valuation, \( w_i(S) = \sum_{q \in D_i} w_i(q) s_{iq} \), otherwise \( w_i(S) = 0 \). The formal definition is as follows.

**Definition 8 (Multi-Unit Combinatorial Auction (\textsc{MUCA}))** Given \((N, M, W, D)\), find an allocation \( S \) of goods to the agents such that \( \sum_{i \in N} \sum_{q \in D_i} w_i(q) s_{iq} \) 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 \( D_i \). Mathematically, \textsc{MUCA} is given by the following integer program (IP):

\[
\begin{align*}
\text{argmax} & \quad \sum_{i \in N} \sum_{q \in D_i} w_i(q) s_{iq} \\
\text{s.t.} & \quad \sum_{i \in N} \sum_{q \in D_i} s_{iq} \leq y(j), \forall j \in \mathcal{G} \\
& \quad \sum_{q \in D_i} s_{iq} \leq 1, \forall i \in N; \quad s_{iq} \in \{0, 1\}, \forall i \in N, \forall q \in D_i
\end{align*}
\]  

(6)

The reduction of \textsc{MIA} to \textsc{MUCA} proceeds as follows. For a given instance \((N, M, V, L, k)\) of \textsc{MIA}, construct an instance of \textsc{MUCA}(\( N, M, W, D \)) problem such that, the set of agents \( N \) 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 \( i \in N \), the demand set \( D_i \) consists of \((m - l_i + 1)\) distinct bundles. Each of the bundles in \( D_i \) 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 \textsc{MIA} from a solution of \textsc{MUCA} in the following way: for every \( q_j \in D_i \) and \( i \in N \), if \( s_{iq_j} = 1 \) then, \( a_{ij} = 1 \) for every \( i \in N \) and \( j \in M \). Similarly, we construct a solution of \textsc{MUCA} from a solution \textsc{MIA} in the following way: if \( a_{ij} = 1 \) for \( i \in N \) and \( j \in M \) then, \( s_{iq_j} = 1 \) for every \( q_j \in D_i \) and \( i \in N \). The following lemma shows that an optimal solution of \textsc{MIA} is an optimal solution of \textsc{MUCA} and vice-versa.

**Lemma 2** Let \( S^* \) is a solution for \textsc{MUCA} for a multiset of goods \( \mathcal{M} \), and \( A^* \) is such that, \( a_{ij}^* = 1 \) for \( i \in N \) and \( j \in M \), if and only if \( s_{iq_j}^* = 1 \) in \( S^* \) for \( i \in N \) and \( q_j \in D_i \), then \( A^* \) is an optimal solution for \textsc{MIA} if and only if \( S^* \) is an optimal solution for \textsc{MUCA}.

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

\[
\sum_{i \in N} \sum_{j \in M} v_{ij} a'_{ij} \geq \sum_{i \in N} \sum_{j \in M} v_{ij} a^*_{ij}
\]

10
The above equation results in a contradiction that implied.

Using Lemma 2 and the next few results, we prove that there exists a polynomial time truthful mechanism individually rational. To find that, we leverage the approximation algorithm of inequality holds, \[ \sum_{i \in N} s_{ij} = 1 \] for every period.

Algorithm 2: \textbf{MAA} in every period

1: Procedure \textbf{MAA}(N, M, V, L, k)
2: \( b \leftarrow \arg \max_{i \in N} v_{ij}; v_{ij} \leftarrow \max_{i \in N, j \in M} v_{ij}; \)
3: \( r \leftarrow \left(6m(k-1)\right)^{\frac{1}{m-2}} \)
4: \( Q^0 \leftarrow \left[0, 0, \ldots, m \right] \) times
5: \( a^*_{bs'} = 1, \) where \( s' \leftarrow \arg \max_{j \in M} v_{bj}; \)
6: \( a^*_{ij} = 0, \forall j \in M \setminus \{s'\}; \)
7: \( P_i \leftarrow \pi_0 \cdot r^{Q_i} \)
8: end for
9: \( a^*_{is} = 1, \) where \( s \leftarrow \max(P_i, v_i) \) (Equation (7))
10: \( a^*_{ij} = 0, \forall j \in M \setminus \{s\} \)
11: \( P_i \leftarrow \sum_{j=s}^{s+l_i-1} P_j \)
12: for \( j = \{1, 2, \ldots, m\} \) do
13: if \( j \in [s, s + l_i - 1] \) then
14: \( Q_j \leftarrow Q_j^1 + 1 \)
15: else
16: \( Q_j \leftarrow Q_j^{j-1} \)
17: end if
18: end for
19: end for
20: return \( a^*, P \)

As \( w_i(q_j) = v_{ij}, \) and \( a^*_{ij} = 1 \) only if \( s_{ij}^* = 1, \) with the constructed \( S' \) corresponding to \( A' \) the following inequality holds,
\[
\sum_{i \in N} \sum_{q_j \in D_i} w_i(q_j) s_{ij}^* \geq \sum_{i \in N} \sum_{q_j \in D_i} w_i(q_j) s_{ij}^*
\]

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

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

Since \textbf{MUCA} is \( \text{NP-complete} \) \cite{13, 29}, using Lemma 2, we get the following theorem.

**Theorem 3** \textbf{MIA} is \( \text{NP-complete}. \)

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 \textbf{MUCA} due to \cite[Theorem 3]{3}. Using Lemma 2 and the next few results, we prove that there exists a polynomial time truthful mechanism (\textbf{MIA} Approximation Algorithm or \textbf{MAA}) to achieve \( O(km^{\frac{1}{m-2}}) \) approximation to the optimal solution of \textbf{MIA}.

The operational principle of \textbf{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\textsuperscript{6} vector which is updated while iterating over the agents in the sequence. We use a superscript \( i \) to denote the the price faced by the agent \( i \) for slot \( j, P_j', \) 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 \textbf{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, \textbf{max} is defined as follows:

\[
\text{max}(v_i, P_i') = \arg \max_{s \in M} \left( v_{is} - \sum_{j \in [s, s + l_i - 1]} P_j' \right)
\]  

\textsuperscript{6}The terms \textit{price} and \textit{delay} are equivalent in the rest of the paper.
MAA maintains a vector \( Q^i = \{Q^i_j\}_{j \in M} \), where \( Q^i_j \) 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 \( Q^i_0 = 0, \forall j \in M \). 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 \( Q^{i-1}_j \), \( r \), and \( \pi_0 \) 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 \( \text{max} \), and the corresponding allocation vector for \( i \) is represented as \( a^*_i \), where \( a^*_{i,j} = 1, a^*_{i,j} = 0, \forall j \neq s \). The total price (or delay) charged to \( i \) is denoted by \( P_i \) and is equal to \( \sum_{j \in [s,s+l_i-1]} P^i_j \). The vector \( Q^i \) 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 \( v_{\max} \).

An important feature of Algorithm 2 is that it does not explicitly check the capacity constraint. However, we show that the choices of \( \pi_0 \) and \( r \) implicitly maintain that in the following result. The units of slot \( j \) after Algorithm 2 executes that are occupied by agents except \( b \) are \( Q^i_j \).

**Lemma 3** Let \( \pi_0, r, \delta > 0 \) be such that \( \pi_0 r^\delta \approx v_{\max} \), then \( Q^i_j \leq \delta + 1 \). This implies that MAA maintains the capacity constraints for each slot for \( \delta = k-2 \).

**Proof**: Assume for contradiction that \( Q^i_j > \delta + 1 \) and let \( i \) be the first customer due to which this contradiction takes place for some slot \( j \), i.e., \( Q^i_j > \delta + 1 \). Since each customer does not get more than one unit of any slot, then it must be that \( Q^{i-1}_j > \delta \). Hence, for slot \( j \), the following holds: \( P^i_j > \pi_0 r^\delta \approx v_{\max} \approx \max_{j \in M} v_{ij} \). 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 \( \text{max} \) since the utility becomes negative for agent \( i \).

The allocation to \( b \) is at most one unit from each slot. With carefully choosing \( \pi_0 \) and \( r \), we bound the units of any slot allocated to all the other agents, \( Q^i_j \) to \((k-1)\), maintaining the possibility of the maximum use of every slot.

Since MAA allocates a slot only if that allocation increases the agent’s utility, the following result holds.

**Theorem 4** MAA is individually rational.

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

**Theorem 5** 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 \( \hat{l}_i < l_i \), \textbf{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 \( P_i \) (which is non-negative). If \( \hat{l}_i > l_i \), then \textbf{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, \textbf{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, \textbf{max} requires at most \(O(m)\) time for every agent \( i \). Therefore, the following result on the complexity of \textbf{MAA} holds.

**Theorem 6** \textbf{MAA} has time complexity \(O(mn)\).

To show the approximation factor of \textbf{MAA}, we need a few more results. We restate a few results from [3], which help us prove the main theorem of Section 6. The lemma and section numbers of these results in the original paper are mentioned within parentheses in the restated lemmata.

**Lemma 4** ([3, Section 4.2, Lemma 4]) For every agent \( i \), \( v_i(a^*_i) \geq v_i(a'_i) - \sum_{j \in [s,s+l_i-1]} \text{s.t. } a'_s=1 P_j^* \) for every allocation \( a'_i \), where, \( P^* \) is the vector of prices of slots at the end of Algorithm 2.\(^7\)

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

**Corollary 3** \( V(ALG^{-b}) \geq V(OPT^{-b}) - (k-1) \sum_{j \in M} P_j^* \)

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

**Lemma 5** ([3, Section 4.2, Lemma 5]) \( V(ALG^{-b}) \geq \sum_{j \in M} P_j^* - \frac{m\pi_0}{r-1} \)

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

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

Following the conditions in Lemma 3 and Lemma 6, we fix \( \pi_0 = \frac{\max_{i \in M} P_{\max}}{6m(k-1)} \), \( r = (6m(k-1))^{\frac{1}{k-2}} \). We restate the following result about the approximation ratio from Bartal et al. [3].

**Lemma 7** ([3, Section 5, Lemma 8]) The approximation ratio of Algorithm 2 is \( 3((k-1)(r-1)+1) \).

Finally, combining Lemmata 6 and 7, we get Theorem 7.

**Theorem 7** 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 \textbf{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.

\(^7\)With a slight abuse of notation, we denote \([a,b]\) to be the integers between \( a \) and \( b \), where \( a < b \).
Multi-slot Divisible jobs

A divisible job implies that the job can be broken up into multiple pieces of unit slot-size and can be executed independently within a period. Partial execution of the jobs is also admissible. Examples of such jobs are parallel computer programs, with each piece being an independent thread of the program. In the context of social scheduling, divisible jobs can be interpreted as independent needs of a customer that may be completed via multiple separate visits to the store without any extra cost. The valuation of the job is the sum of the valuations of the allocated slots. We assume that the length $l_i$ of the job is verifiable and is common knowledge. Hence, the valuations of each of the slots, i.e., $v_{ij}$’s, are the agents’ only private information. In this setting, the valuation of agent $i$ from the allocation $A$ can be written as $v_i(A) = \sum_{j:a_{ij}=1} v_{ij}$. Note that the EPP condition under this setting is identical to the LP (3) with the first set of constraints replaced by $\sum_{j \in M} a_{ij} \leq l_i$, $\forall i \in N$. However, this does not alter the constraint matrix $C$, which is TU (shown in the proof of Theorem 1). Therefore the new LP also has integral optimal solution. The modified VCG-T is identical to Algorithm 1 with Step 2 replaced with the solution to the modified LP as explained above. Hence, the following results hold similarly.

**Proposition 1** Modified VCG-T is per period dominant strategy truthful for multi-slot divisible jobs.

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

**Theorem 8** The allocation of the modified VCG-T always gives integral solutions.

Hence, the following proposition also holds.

**Proposition 3** Modified VCG-T is EPP.

To reduce the allocation problem to a perfect $b$-matching problem, we need to alter $b_i = l_i$, and Lemma 1 holds. Therefore, we conclude the following result.

**Theorem 9** Modified VCG-T provides a combinatorial, strongly polynomial algorithm for computing a social schedule and delays.

8 Experiments

While the mechanisms presented in the previous sections 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 (§8.1).

For multi-slot jobs, we simulated MAA to find the suboptimality and the reduction in the running time (§8.2).

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. The valuations for the other slots are assumed to decrease with a multiplicative factor $\delta \in (0, 1)$ in the order of their slot preference, i.e., the valuation for the $t$-th most preferred slot of a medium agent will be $2\delta^{t-1}$. We used Gurobi [20] under an academic license for all experiments.
8.1 Reduction in the social congestion

This section considers a real data of customer footfall in a general store that we have collected from the store. 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 28 (top) and 32 (bottom).]

We divide the store opening hours into 14 hourly slots between 7 AM to 9 PM. The valuations of every customer is drawn from a distribution \{\text{high:0.1, medium:0.3, low:0.6}\}. In this experiment, an agent who is not allocated any slot under VCG-T on a day is removed from the system and counted separately. Fig. 1 shows the comparison of the average current population with that allocated by VCG-T for slot capacities of 28 (above) and 32 (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. The rows of Fig. 2 show the allocated slot preference and the delays for the three difference classes of valuations for the slot capacities 28 and 32 respectively. It shows that a higher valuation comes with a higher delay.

8.2 Suboptimality and Complexity Reduction (MAA)

The sub-optimality of MAA (Algorithm 2) was obtained for a worst-case scenario in §6. Here we investigate the sub-optimality of MAA and the amount of time it reduces w.r.t. an algorithm that finds the optimal allocation of the slots. The top plot of Fig. 3 shows the percentage reduction \((t_{\text{OPT}} - t_{\text{MAA}})/t_{\text{OPT}}\) in the running time of MAA compared to the optimal mechanism, where \(t_{\text{OPT}}\) and \(t_{\text{MAA}}\) 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**. The experiment is run with \( n = 6, k = 5, \) and \( m \) varying from 3 to 8. For each value of \( m, n, k, \) the experiment is repeated 100 times. The parameters 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**.](image1)

![Figure 4: Priority-delay trade-off for **VCG-T**.](image2)

### 8.3 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 Fig. 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.

However, **VCG-T** does the prioritized allocations of the agents at the cost of their delays. The bottom plot of Fig. 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.

### 8.4 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. Fig. 5 shows the growth of the computation time of the mechanism. As a reference, to solve the allocation and delays for the store of Section 8.1, 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.

16
Figure 5: Computation times of VCG-T.

Epilogue: This paper provides a solution to social distancing using social scheduling keeping the COVID-19 pandemic as a motivation, but the solution apply to more general settings too. We have already developed an app that runs VCG-T.
References

[1] R. Arazi and A. Feigel. Discontinuous transitions of social distancing in the sir model. *Physica A: Statistical Mechanics and its Applications*, 566:125632, 2021.

[2] J. C. Aydos and A. O’Brien. Scats ramp metering: strategies, arterial integration and results. In *17th international IEEE conference on intelligent transportation systems (ITSC)*, pages 2194–2201. IEEE, 2014.

[3] Y. Bartal, R. Gonen, and N. Nisan. Incentive compatible multi unit combinatorial auctions. In *Proceedings of the 9th conference on Theoretical aspects of rationality and knowledge*, pages 72–87, 2003.

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

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

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

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

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

[9] R. Cavallo. Social distancing equilibrium. Unpublished, 2021. URL http://www.eecs.harvard.edu/~cavallo/wp/cavallo-socdisteq.pdf.

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

[11] S. Cho. Mean-field game analysis of sir model with social distancing. *arXiv preprint arXiv:2005.06758*, 2020.

[12] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

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

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

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

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

[17] M. Gosak, M. U. Kraemer, H. H. Nax, M. Perc, and B. S. Pradelski. Endogenous social distancing and its underappreciated impact on the epidemic curve. *Scientific reports*, 11(1):1–10, 2021.
[18] M. Grötschel, L. Lovász, and A. Schrijver. Complexity, oracles, and numerical computation. In *Geometric Algorithms and Combinatorial Optimization*, pages 21–45. Springer, 1993.

[19] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[20] L. Gurobi Optimization. Gurobi optimizer reference manual, 2020. URL http://www.gurobi.com.

[21] M. T. Hajiaghayi. Online auctions with re-usable goods. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 165–174, 2005.

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

[23] R. Lavi and N. Nisan. Competitive analysis of incentive compatible online auctions. *Theoretical Computer Science*, pages 310:159–180, 2004.

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

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

[26] B. Pejó and G. Biczók. Corona games: Masks, social distancing and mechanism design. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Modeling and Understanding the Spread of COVID-19*, pages 24–31, 2020.

[27] T. C. Reluga. Game theory of social distancing in response to an epidemic. *PLoS Comput Biol*, 6(5): e1000793, 2010.

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

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

[30] M. Roycroft, N. B. Bhala, S. Brockbank, D. O’Donoghue, A. Goddard, and A. M. Verma. Rostering in a pandemic: Sustainability is key. *Future Healthc J*, 2020.

[31] S. K. Sahni. Algorithms for scheduling independent tasks. *J. ACM*, 23(1):116–127, Jan. 1976. ISSN 0004-5411. doi: 10.1145/321921.321934. URL https://doi.org/10.1145/321921.321934.

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

[33] M. Shadmehr and E. B. de Mesquita. Coordination and social distancing: Inertia in the aggregate response to covid-19. *University of Chicago, Becker Friedman Institute for Economics Working Paper*, (2020-53), 2020.

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

[35] L. Thunström, S. C. Newbold, D. Finnoff, M. Ashworth, and J. F. Shogren. The benefits and costs of using social distancing to flatten the curve for covid-19. *Journal of Benefit-Cost Analysis*, pages 1–27, 2020.
[36] F. M. Toxvaerd. Equilibrium social distancing. 2020.

[37] W. Vickrey. Counter speculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1): 8–37, 1961.

[38] A. Wilder-Smith and D. O. Freedman. 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*, 2020.