Prior-free Strategic Multiagent Scheduling with focus on Social Distancing

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

[1]Indian Institute of Technology Kanpur, {deepeshk,garimas,swaprava}@iitk.ac.in, * = equal contribution

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). The algorithm takes input from citizens and schedules the store's time-slots based on their importance to visit the facility. Naturally, the formulation applies to several similar problems. We consider the single and multiple slot demands of the requests. 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 users' 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. We show that the problem becomes NP-complete as soon as the multi-slot demands are *indivisible*, and provide a polynomial-time mechanism that is truthful, individually rational, and approximately optimal. 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 it reduces the social congestion significantly using users' visit data from a store. For the multi-slot indivisible jobs, our mechanism yields reasonable approximation while reducing the computation time significantly.

Keywords: social distancing, social scheduling, strong polynomial, mechanism design, poly-time approximation.

1 Introduction

Pandemics like COVID-19 showed that the most effective solution against infectious diseases is *social distancing* [36]. Therefore, it is a practice we ought to master and stay prepared as a preventive disease containment measure in the future. While 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, the customers have different levels of importance and independently decide their moves. We assume that they are *rational*, i.e., along with their importance, they are also concerned with their health safety and want to maintain a social distance.

In this paper, we consider *three* settings where the task of visiting the store takes (i) a single time-slot, (ii) multiple slots–but can be non-contiguous and the task can be partially executed, and (iii) multiple contiguous slots and the task is not partially executable (we call it an *indivisible* task).

For cases (i) and (ii), we propose `IMPPreSS` (Incentive Mechanism for Priority Preserving Social Scheduling), a mechanism inspired by the Vickrey-Clarke-Groves (VCG) mechanism [10, 18, 35], which solves this social coordination problem with *mechanism design.* The mechanism is run for a facility that has a recommended maximum footfall (the *capacity*) for a duration of time (a *slot*) to maintain social distance. Unlike various other appointment maintaining apps, `IMPPreSS` takes inputs from all users willing to visit the store and schedules them according to their importance and slot preferences. The algorithm is capable of (a) ensuring that the users do not overstate (or understate) their importance, (b) maintaining voluntary participation of the users, which are expected since it is inspired from the VCG. However, it also (c) can schedule a large number of users to a large number of slots in a computationally tractable manner.

For case (iii), we show that the problem is computationally intractable and propose a different mechanism `MAA` (`MULTI-INDIV-ALLOC Approximation Algorithm`) that is truthful, approximately optimal, ensures voluntary participation, and is computationally tractable.

Though cast in the context of social scheduling for pandemics, a similar problem arises in general scheduling settings, e.g., scheduling traffic in freeways (ramp metering[1]) or multi-ownership jobs in a single processor (multiprocessor scheduling[2]). Since all such settings have multiple agents competing for a common resource, the solutions involving truthful revelation in a computationally tractable manner apply to those settings as well. 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.[3] A valuation v_{ij} denotes the maximum number of slots agent i is willing to wait before requesting the facility again for the next unprocessed job if her current job is allocated the slot j. A higher value indicates that she has higher importance for that slot. The valuations of a customer (agent) are represented by a vector of real numbers, whose length is the number of slots in that period.[4] A multiagent scheduling mechanism takes these reports, allocates the slots in that period, and decides a delay for each agent. The delay is the time duration that captures till when an agent cannot access the facility starting from the time of the allocated slot. Quite naturally, an agent prefers to have a more valuable slot assigned to her with less delay. This competitive scenario induces a *game* where agents' actions are to report the valuations. The agents may *overstate* (or *understate*, both of which are different from truthtelling) their actual valuations.

We model the agents' payoffs using the well-known quasi-linear payoff model [32, Chap 10] and propose `IMPPreSS`, which is inspired by the VCG allocation and transfers–the transfer is set as the delay in every period of this game (for the single slot problem that considers the setting where every agent has a job length of one slot). The allocation ensures that each slot admits a maximum number of agents that maintains the social distancing norms. The multi-slot problem considers the scenario where the jobs can take multiple slots – however, it may either be divisible, i.e., the total job may not need contiguous time-slots, or indivisible, i.e., needs contiguous time-slots. Note that the mechanisms presented in this paper do not depend on the priors of the agent valuations.[5] We show that `IMPPreSS` inherits certain known properties of VCG, e.g., optimality, truthfulness, and participation guarantee in single slot and divisible multi-slot cases. However, even though the allocation and delay computation involves solving combinatorial optimization problems, we prove that it is strongly polynomial. Moreover, the indivisible multi-slot job scheduling problem is NP-complete and we provide a polynomial-time approximately efficient mechanism (`MAA`) that satisfies truthfulness and participation guarantees.

Table 1 summarizes the theoretical contributions made in this paper. We list the inherited properties as propositions and the main contributions as theorems.

[1] `https://en.wikipedia.org/wiki/Ramp_meter`

[2] `https://en.wikipedia.org/wiki/Multiprocessor_scheduling`

[3] The type of demands we consider in this paper, e.g., shopping needs, has a recurrent nature and the importance of such a job is not known apriori as it depends on that time, possible previous non-allocation of slots, missed item during previous shopping instance, etc.

[4] The valuation representation of the slots changes slightly when we consider multi-slot indivisible jobs and is detailed in Section 6.2.

[5] An extension of our model allows dynamic arrivals where the planner needs to decide to allocate the slots in an online manner. However, it needs the priors of the valuations. We leave that extension for a future investigation.

Table 1: Summary of the theoretical results

Properties ↓ **Jobs** →	Single-slot	Multi-slot (divisible)	Multi-slot (indivisible)
Efficiency	Optimal (Proposition 3)	Optimal (Proposition 6)	Approximately Optimal, $O\left(km^{\frac{1}{k-2}}\right)$ (Theorem 9)
Truthfulness	Yes (Proposition 1)	Yes (Proposition 4)	Yes (Theorem 7)
Participation guarantee	Yes (Proposition 2)	Yes (Proposition 5)	Yes (Theorem 6)
Computational tractability	Strongly polynomial (Theorems 1 and 2)	Strongly polynomial (Theorems 3 and 4)	Polynomial, $O(mn)$ (Theorem 8)

Experiments (Section 7) 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 (Section 7.1). We show that it reduces the social congestion significantly using users' visit data from a store (Section 7.2). For the multi-slot indivisible jobs, our approximately optimal mechanism provides a reasonable approximation at a much reduced computational cost in practice (Section 7.4).

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

- We adapt the VCG mechanism for the single-slot and divisible multi-slot jobs where the transfers are the delays (cooling-off time) for the agents and show that efficiency, truthfulness, and participation can be guaranteed.
- We prove that even though this is an integer programming problem, the solution is actually computable in strongly polynomial time.
- For indivisible multi-slot jobs we propose an approximately optimal polynomial time mechanism which ensures truthfulness and participation.
- Our real and synthetic data experiments show that the congestion can be greatly reduced (thereby social distancing improved) in real stores. Also, for indivisible multi-slot jobs, the mechanism yields a not very large approximation ratio at a large computational gain.

1.2 Related Work

Social distancing measures have been widely successful and recommended for pandemic control [11, 15, 33]. 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 is to understand the relationship of social distancing with different public policies and other factors [1, 9, 16, 34]. Pejó et al. [25] model the coronavirus situation as a game between rational agents where the strategies are whether to wear a mask or not, to go out for a meeting or not, and others. They show how the decision of an agent to go out to attend a meeting at the risk of her own life depends on the maximum duration, the maximum number of people allowed in a gathering, and the probability of getting infected. Toxvaerd [34] considers social distancing in the susceptible-infected-recovered (SIR) epidemiological model and characterizes the equilibrium dynamics in terms of peak prevalence, time-domain properties of the disease, and other factors. Toxvaerd [34] shows that "during the equilibrium social distancing phase, individuals gradually reduce their social distancing efforts despite the infection probability not decreasing."

Individuals are sometimes reluctant to pay the costs inherent in social distancing which can limit its effectiveness as a control measure [26]. This is particularly important when the movement of the people is largely uncoordinated. Under an uncoordinated model, Cavallo [7] shows that every equilibrium involves more social contact than it occurs in a social optima. Coordination is essential for social distancing [31], 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.

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 [28]. However, these measures are centrally controlled, and an individual does not have a scope to behave strategically.

On the other hand, the businesses maintain social distancing by enforcing certain appointment booking apps (either web or smartphone-based) to their customers. These apps run a *first-come-first-served* (FCFS) algorithm for slot booking and do not provide any priority preservation guarantees.[6]

Resource allocation with monetary transfers to ensure truthfulness, e.g., in the context of jobshop scheduling, has a rich literature that is close to our work (even though our mechanisms do not use money). Friedman et al. [12] consider interruptible and non-interruptible versions of the online *single* resource allocation problem, where the resources can be allocated to a limited number of users, e.g., the WiFi at Starbucks or shared computation on a server. They assume that the agents can not lie about their arrival time and get better valuations and prove that the online version of VCG is strategyproof only if the allocation rule is time-monotonic. They also show that a simple fixed price payment can result in a close-to-optimal solution for a non-interruptible version of their problem. Lavi and Nisan [22] 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). Hajiaghayi et al. [20] provide an online auction for reusable goods, e.g., assigning time slots of a machine to unit length jobs. They study *single-capacity machines* and give a truthful randomized mechanism. Our work is different in the sense that the jobs we consider and the machine capacities can be of varied length. Chen et al. [8] consider a job allocation to machines which is closely related to our setting. However, the job arrivals and allocation are online in that setting. They propose a truthful approximate mechanism for models 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 efficient allocation truthfully and also ensures budget balance [4, 13, 24], while our model can admit costs of any structure.

In the context of job scheduling without money, Koutsoupias [21] studies the allocation of independent tasks to machines, where the machines are lazy agents and prefer not to execute any task. Every machine reports the time it takes to execute each task (private information), and the allocation is done to minimize the *makespan*. The author provides a truthful in expectation, approximate mechanism without money for one task—which can be repeated for multiple tasks. In our setup, we consider maximizing the *sum of the visitors' valuations* which are independent of the length of the job and provide an approximate mechanism for multiple tasks respecting the capacity at all times.

Our problem formulation and its solution use time as a replacement for money in the context of pandemic containment. While social distancing is a highly desirable coordinated effort, charging money to do the same is perhaps unethical and unacceptable in many countries. However, waiting time is often seen as a resource that individuals agree to trade with [23]. Hence, our approach will work in all places where payment can be replaced with a delay in time.

2 Basic Problem Setup

Define $N \coloneqq \{1, \ldots, n\}$ 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 $M \coloneqq \{1, \ldots, m\}$. Every slot has a maximum capacity of k, which is decided by the region's social distancing norm and the size of the facility. A *central planner* (e.g., an AI app) allocates these slots to the agents, maintaining the capacity constraint. The capacity of each slot is k, which is an input to the allocation problem and is decided beforehand (e.g., by medical professionals for a safe social distance in a specific store size).[7] Every agent has a *cardinal* preference, which is represented by a real number against each slot, over the slots. For instance, the preference (we will call this the agent's *valuation*) of agent i for slot j is denoted by $v_{ij} \in \mathbb{R}_{\geqslant 0}$, which is

[6] An example app is www.appointy.com.

[7] The analysis and results will follow even if the capacity k_j varies with the slots $j \in M$. We kept it a constant since in our setting, the safe socially distancable population cannot change with time for the same store.

interpreted as if slot j is allotted to i, then i is willing to wait for at most v_{ij} time periods before requesting another access to the facility.[8] Thus, 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 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 a 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. We denote the slot assigned to i by a^*_i. The planner also decides a delay $d = (d_i, i \in N)$ for each agent before which they cannot make another request to the system. We assume that every agent wants a more valued slot to be assigned to her and also wants to wait less. Therefore, the net payoff of an agent is assumed to follow a standard *quasi-linear form* [32]

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

Denote the set of all allocations by $\mathcal{A}$. The delays $d_i \in \mathbb{R}, \forall i \in N$. The planner does not know the valuations (or importances) 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, e.g., an agent can report a `not-urgent` valuation (lower v_{ij}) as `urgent` (higher v_{ij}) for a prioritized scheduling. To distinguish, we use v_{ij} for the true valuation and $\hat{v}_{ij}$ for reported valuations. We will 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.*[9]

3 Preliminary Definitions

In this section, we formally define a few desirable properties that a social scheduling algorithm 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*

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

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.

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* $i \in N$

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

[8]The valuation v_{ij} need not be integral. In practice, the facility may be frozen for a maximum duration equal to that number, which is considered as the *cooling-off* time in the booking app.

[9]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.

The inequality above shows that if the true value of agent i is v_i, the allocation and delay resulting from reporting it 'truthfully' maximizes her payoff *irrespective of the reports of the other agents.*

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

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

Individual rationality ensures that it is always weakly beneficial for every rational agent to participate in such a mechanism.

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.* For completeness of the paper, we explain this class of problems.

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 **5** (Strongly Polynomial)**.** *An algorithm runs in* strongly polynomial time *if [17]*

- *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*
- *the space used by the algorithm is bounded by a polynomial in the input size.*

The SSF is strongly polynomial time computable if there exists an algorithm which 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.

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.[10]

In our setting, the agents have the opportunity to overstate their importances to get a better slot allotted to them. This is precisely where our approach using the ideas of mechanism design [5] is useful.

Algorithm 1 `IMPPreSS` in every period

```
1: Input: for every agent $i \in N$, the value $\hat{v}_i$
2: compute $A^*(\hat{v})$ (Equation (4)) as the allocation
3: charge a delay $d_i(\hat{v})$ (Equation (5)) to every $i \in N$
   for which they cannot access the scheduling mechanism again
4: Output: $A^*(\hat{v})$ and $d(\hat{v})$
```

In a setting where the agents' preferences are private, if the mechanism has no additional structures (e.g., allows delays in time in our context), only *dictatorial* mechanisms are truthful [14, 29]. This negative result holds irrespective of whether agents' preferences are *ordinal* (representable as an order relation over the outcomes) or *cardinal* (agents have a real number to represent the intensity of the preference). Note that in our setup, the agents' preferences are cardinal. A complementary analysis by [27, Thm 7.2] shows that a dictatorship result reappears under certain mild conditions in a *quasi-linear* setting (which is our current setting) unless transfers (of utility) are allowed. Therefore, use of transfers in some form is inevitable to ensure truthfulness of the agents. In this paper, we use the delay as a surrogate for transferable utility among the agents to ensure truthful value revelation at every period. We present the mechanism `IMPPreSS` (`I`ncentive `M`echanism for `P`riority `Pre`serving `S`ocial `S`cheduling) as follows.

[10]For mechanisms that consider the dynamic extension of such allocation problems with finite or infinite horizon [3, 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.

Description of `IMPPreSS`. The SSF needs to decide on the allocation A and the delay d. The actual allocation that satisfies EPP is given by the following integer program (IP).

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

However, IPs are NP-complete in general. Therefore, `IMPPreSS` computes the LP relaxation of the above problem, given as follows.

$$\begin{aligned} &\underset{A}{\operatorname{argmax}} \sum_{j \in M} \sum_{i \in N} v_{ij} a_{ij} \\ \text{s.t.} \ \sum_{j \in M} a_{ij} \leqslant 1, \ \forall i \in N, \quad &\sum_{i \in N} a_{ij} \leqslant k, \ \forall j \in M, \quad a_{ij} \geqslant 0, \ \forall i \in N, j \in M. \end{aligned} \tag{4}$$

We will show that this is without loss of optimality since the solution to LP (4) will always be integral and will coincide with the solution of IP (3).

The delays of agents are computed via the standard Vickery-Clarke-Groves (VCG) rule [10, 18, 35]. Denote the optimal allocation of LP (4) by $A^*(v)$. Also, denote the allocation given by LP (4) when agent i is removed from the system by $A^*_{-i}(v_{-i})$. For agent i, the delay is given by,

$$d_i := \sum_{\ell \in N \setminus \{i\}} v_\ell(A^*_{-i}) - \sum_{\ell \in N \setminus \{i\}} v_\ell(A^*). \tag{5}$$

The mechanism in every period is described in Algorithm 1.

Note that the allocation is different from that of a *sorted importance sequential dictator*, i.e., where the agents are sorted based on their importance and asked to pick their favorite slots in that order. For example, consider two agents, A and B, and two slots of capacity one. Valuations are (51, 50) and (50, 0) respectively for A and B respectively for the two slots – sequential dictator would allocate slot 1 to A and 2 to B, while the socially optimal is to allocate slots 2 and 1 to A and B respectively, which is allocated by `IMPPreSS`.

5 Theoretical guarantees

Our first result shows that `IMPPreSS` reveals their importance truthfully.

PROPOSITION **1.** `IMPPreSS` *is per period dominant strategy truthful.*

This proof of per period dominant strategy truthful is a standard extension of the proof for Vickery-Clarke-Groves (VCG) mechanism [10, 18, 35] and is available in Appendix A.

The following result ensures that the agents are incentivized to voluntarily participate in `IMPPreSS`.

PROPOSITION **2.** `IMPPreSS` *is individually rational for every agent.*

The proof of the above result is deferred to the Appendix B.

The next few results show that even if the allocation problem of IP (3) falls in a computationally intractable class, its special structure can be used to find a tractable solution.

THEOREM **1.** *The allocation of* `IMPPreSS` *given by LP (4) 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 (4) 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 (4) respectively. We show that C is totally unimodular (TU), which is sufficient to conclude that LP (4) has integral solutions. We use the Ghouila-Houri characterization [6] 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 (4) is an optimal solution of IP (3). Hence, the mechanism is indeed doing an efficient allocation which can be summarized in the following proposition.

PROPOSITION **3.** `IMPPreSS` *is EPP.*

Theorem 1 shows that the solution to the LP relaxation of the allocation problem is without loss of optimality. LPs are known to be polynomially computable. However, in general, it 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. The forthcoming results show that the solutions of allocation and delays of `IMPPreSS` are strongly polynomial (Definition 5). To show this, we will first reduce the allocation problem (LP (4)) to a minimum weight b-matching problem, which is known to be strongly polynomial [30].

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 (4).*

Proof. We prove this via contradiction. Suppose A^* is not an optimal solution to LP (4), 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 (4), 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 (4) 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 **1.** *The computation of the delays in* `IMPPreSS` *is strongly polynomial.*

Combining Theorem 1, Lemma 1, and Corollary 1, we get the following result.

THEOREM **2.** `IMPPreSS` *provides a combinatorial, strongly polynomial algorithm for computing a social schedule and delays.*

The following section addresses the problem with every agent having jobs with duration spanning multiple time-slots.

6 Multi-slot Jobs

In this section, we consider jobs with different lengths, i.e., for agent i, the job may be of length $l_i \geqslant 1$. We consider two cases of such jobs, which can be either *divisible* or *indivisible*.

6.1 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 $v_i(A) = \sum_{j:a_{ij}=1} v_{ij} a_{ij}$. Note that the EPP condition under this setting is identical to the IP (3) with the first set of constraints replaced by $\sum_{j \in M} a_{ij} \leqslant l_i, \ \forall i \in N$. The LP relaxation of the new IP is identical to LP (4) with the first set of constraints replaced with the above set of constraints. However, this does not alter the constraint matrix C, which is TU (shown in the proof of Theorem 1). Therefore the new LP also gives integral solutions, which is an optimal solution to the new IP. The *modified* `IMPPreSS` 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 **4.** *Modified* `IMPPreSS` *is per period dominant strategy truthful for multi-slot divisible jobs.*

PROPOSITION **5.** *Modified* `IMPPreSS` *is individually rational for every agent.*

THEOREM **3.** *The allocation of the modified* `IMPPreSS` *always gives integral solutions.*

Hence, the following proposition also holds.

PROPOSITION **6.** *Modified* `IMPPreSS` *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 **4.** *Modified* `IMPPreSS` *provides a combinatorial, strongly polynomial algorithm for computing a social schedule and delays.*

6.2 Indivisible jobs

A job is *indivisible* if the entire length l_i of the job requires contiguous slots for execution within the period. Examples of such jobs are computer programs that are not parallelizable and need continuous execution. In the social scheduling setting, an individual may not be willing to visit the store multiple times, and therefore the allocation needs to provide contiguous time-slots to that agent. We show that the optimal allocation problem in such a setting can be computationally intractable.

Each agent i comes with a job having a valuation v_{ij} (private) if her job begins at slot j, and its length l_i (public). 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 agents' valuations, and L consists of the lengths of agents' jobs. An allocation is a matrix $\mathcal{A} = [\mathfrak{a}_{ij}]$, where $\mathfrak{a}_{ij} = 1$ if agent i's job starts at slot j, else $\mathfrak{a}_{ij} = 0$. By following all other notations as before, we define the Multi-slot Indivisible Jobs Allocation problem as:

Definition **6.** ***Multi-slot Indivisible Jobs Allocation Problem*** *(*`MULTI-INDIV-ALLOC`*): Given (N, M, V, L, k), find an allocation $\mathcal{A}$, 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,* `MULTI-INDIV-ALLOC` *is given by the following integer program (IP).*

$$\underset{A}{\operatorname{argmax}} \sum_{i\in N}\sum_{j\in M} v_{ij}\ \mathfrak{a}_{ij}$$
$$s.t.\ \sum_{i\in N}\ \sum_{\substack{p\in M\\ j\leqslant p+l_i-1}} \mathfrak{a}_{ip} \leqslant k, \forall j \in M, \quad \sum_{j\in M} \mathfrak{a}_{ij} \leqslant 1, \forall i \in N, \quad \mathfrak{a}_{ij} \in \{0,1\}, \forall i \in N, \forall j \in M \tag{6}$$

Depending on the opening hours of the facility and the duration of each slot, $|M|$ can be a small integer. When the input variables such as the number of slots and number of customers are small, `MULTI-INDIV-ALLOC` can be solved in polynomial time by simply checking for every possible solution and choosing the optimal among them. However, for completeness and to avoid situations where our results can not be applied, we consider the computationally expensive conditions for analysis. For example, the size of the period can be significantly longer; hence, $|M|$ can be a large number.

Our following result shows that `MULTI-INDIV-ALLOC` is computationally intractable.

Theorem **5.** `MULTI-INDIV-ALLOC` *is NP-complete.*

The above theorem is proved using the reduction from the knapsack problem and the proof is available in Appendix C.

Though finding the efficient allocation for `MULTI-INDIV-ALLOC` is intractable (Theorem 5), it is possible to find an approximately efficient allocation in polynomial time that is truthful and individually rational. In the following, we present a truthful polynomial time algorithm to achieve $O(k\ m^{\frac{1}{k-2}})$ approximation to the optimal solution for `MULTI-INDIV-ALLOC` problem, by reducing it to Multi-Unit Combinatorial Auction (`MUCA`) problem.

Consider a multiset $\mathcal{M} = (\mathcal{G}, y)$, where $\mathcal{G} = \{1, 2, 3, \dots, g\}$ 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 $i \in \mathcal{N} = \{1, 2, \dots, n\}$ 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 $w_i(\mathcal{S}) = 0$. `MUCA` is defined as follows:

Definition **7** (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 the availability of j, $y(j)$, and every agent i is assigned exactly one of the demanded bundle from $\mathcal{D}_i$. Mathematically,* `MUCA` *is given by the following integer program (IP):*

$$\underset{\mathcal{S}}{\operatorname{argmax}} \sum_{i\in\mathcal{N}}\sum_{q\in\mathcal{D}_i} w_i(q)\ \mathfrak{s}_{iq}$$
$$s.t.\ \sum_{i\in\mathcal{N}}\sum_{\substack{q\in\mathcal{D}_i\\ j\in q}} \mathfrak{s}_{iq} \leqslant y(j), \forall j \in \mathcal{G}, \quad \sum_{q\in\mathcal{D}_i} \mathfrak{s}_{iq} \leqslant 1, \forall i \in \mathcal{N}, \quad \mathfrak{s}_{iq} \in \{0,1\}, \forall i \in \mathcal{N}, \forall q \in \mathcal{D}_i \tag{7}$$

The reduction of `MULTI-INDIV-ALLOC` to `MUCA` proceeds as follows. For a given instance (N, M, V, L, k) of `MULTI-INDIV-ALLOC`, construct an instance of $\texttt{MUCA}(\mathcal{N}, \mathcal{M}, W, \mathcal{D})$ 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} (valuation $i \in N$ gets if her job starts at slot $j \in M$). The above construction can be done in polynomial time of input size.

We construct a solution of `MULTI-INDIV-ALLOC` 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 `MULTI-INDIV-ALLOC` 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 `MULTI-INDIV-ALLOC` is an optimal solution of `MUCA` and vice-versa.

Lemma **2.** *Let $\mathcal{S}^*$ is an optimal solution for* `MUCA` *for a multiset of goods $\mathcal{M}$, and $\mathcal{A}^*$ 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* `MULTI-INDIV-ALLOC`.

Proof. Suppose the above statement is not true and hence $\mathcal{A}^{'}$ but not $\mathcal{A}^*$ is an optimal solution for `MULTI-INDIV-ALLOC`.

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

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 $\mathcal{A}^{'}$ 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.

To prove the other direction, we construct $\mathcal{A}^*$ for `MULTI-INDIV-ALLOC` from the optimal solution $\mathcal{S}^*$ for `MUCA` as explained before the Lemma 2. The proof follows by similar argument in the reverse order. $\square$

Since there exists a polynomial time truthful algorithm to achieve $O\big(kg^{\frac{1}{k-2}}\big)$ approximation to optimal solution for `MUCA` problem [2, Theorem (3)]; from the above lemma and next few results, we prove that there exists a polynomial time truthful algorithm to achieve $O\big(km^{\frac{1}{k-2}}\big)$ approximation to optimal solution for `MULTI-INDIV-ALLOC` problem.

We present a mechanism `MULTI-INDIV-ALLOC` Approximation Algorithm (`MAA`) that approximates the optimal solution for `MULTI-INDIV-ALLOC` and is described in Algorithm 2. The mechanism comes with a price[11] vector P of length m, where P_j represents the price of slot j. The mechanism also uses a `Demand-Oracle` which takes into account agent i's valuation function and the price vector P and returns the slot s such that the utility of the agent i is maximum if she gets slots s to $(s + l_i - 1)$. Therefore, it is a utility maximizer slot allocator for i. The operational principle of `MAA` is a sequential dictatorship, where the sequence is an arbitrary order of the agents and is independent of the information submitted by them. This is explained as follows.

The mechanism finds the agent b who has maximum valuation for any slot, sets a rate factor $r = (6m(k-1))^{\frac{1}{k-2}}$ (the use of this factor is explained later), and maintains a vector $\mathcal{Q} = [\mathcal{Q}_j]_{j \in M}$ such that, $\mathcal{Q}_j$ denotes the units of slot j already allocated to agents. Initially, $\mathcal{Q}_j = 0$ and the price of slots $P_j = \frac{v_{\max}}{6m(k-1)}$ for every $j \in M$, where $v_{\max}$ is the maximum valuation among all agents and slots. Consider an arbitrary order (WLOG $(1, 2, \ldots, n)$) of the agents. Each agent $i \neq b$ faces a price vector $P^i = (P^i_j, j \in M)$, which denotes the prices of all slots. The `Demand-Oracle` finds a slot s such that the utility of i is maximum (w.r.t. P^i and $v_i(\cdot)$) if she gets the contiguous slots s to $(s + l_i - 1)$. The payment made by i is denoted by $\mathcal{P}_i$ and is equal to $\sum_{j=s}^{s+l_i-1} P^i_j$. The vector $\mathcal{Q}$ is updated after the allocation of slots to agent i. The price vector P

[11]Price and delay imply the same in our context.

is also updated such that the prices of the slots increase by a multiplicative factor of a suitable exponent of $r(> 1)$ such that the prices for more demanded slots are higher. The agent b gets her most valued starting slot and pays the maximum valuation among all other agents and slots – this is represented by $v_{\max}^{-b}$. We present the details of the mechanism in Algorithm 2. Let $\mathcal{Q}_j^*$ represent the units of slot j allocated to all the customers except b after Algorithm 2 executes. The result that we present below is similar to that in [2, Lemmata 2 and 7], where P^0 and r are as defined in Algorithm 2.

Algorithm 2 MAA (MULTI-INDIV-ALLOC Approximation Algorithm) in every period

```
1: **Procedure** MAA$(N, M, V, L, k)$
2: $b \leftarrow \arg_i \max_{i \in N, j \in M} v_{ij}$
3: $v_{\max} \leftarrow \max_{i \in N, j \in M} v_{ij}$
4: $r \leftarrow (6m(k-1))^{\frac{1}{k-2}}$
5: $\mathcal{Q}(1) \leftarrow [0\ 0\ \ldots\ m\ times]$
6: **for** $j \in M$ **do**
7:     $P_j^0 = \frac{v_{\max}}{6m(k-1)}$
8: **for** $i \in N$ **do**
9:     $v_{\max}^{-i} \leftarrow \max_{t \in N \setminus \{i\}, j \in M} v_{tj}$
10:    **if** $i \neq b$ **then**
11:        **for** $j \in M$ **do**
12:            $P_j^i \leftarrow P_j^0 \cdot r^{[Q(i)]_j}$
13:        $\mathfrak{a}_i^* \leftarrow$ Demand-Oracle$(P^i, v_i(.))$
14:        $\mathcal{P}_i \leftarrow \sum_{j \in M,\ j \in [s, s+l_i-1],\ \mathfrak{a}_{is}^*=1} P_j^i$
15:        **for** $j \in M$ **do**
16:            **if** $j \leqslant s + l_i - 1$ *where* $\mathfrak{a}_{is}^* = 1$ **then**
17:                $[\mathcal{Q}(i+1)]_j \leftarrow [\mathcal{Q}(i)]_j + 1$
18:            **else**
19:                $[\mathcal{Q}(i+1)]_j \leftarrow [\mathcal{Q}(i)]_j$
20:    **else**
21:        $\mathfrak{a}_b^*, \mathcal{P}_b \leftarrow \text{argmax}_{j \in M}(v_{bj}),\ v_{\max}^{-b}$
22: **return** $\mathfrak{a}, \mathcal{P}$
```

LEMMA **3.** *Let P^0, r be such that $P^0 r^\delta \geqslant v_{\max}$, then $\mathcal{Q}_j^* \leqslant \delta + 1$. In particular,* MAA *maintains the capacity constraints for each slot for $\delta = k - 2$.*

Proof. The allocation to b is at most *one* unit from each slot (good). With carefully choosing P^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.

To prove the lemma, 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., $\mathcal{Q}_j(i+1) > \delta + 1$. Since each customer does not get more than one unit of any slot, then it must be that $\mathcal{Q}_j(i) > \delta$. Hence, for slot j, the following holds: $P_j^i > P^0 r^\delta \geqslant v_{\max} \geqslant \max_{j \in M}\ v_{ij}$. This makes i's total payment of l_i contiguous slots including slot j to be more than her corresponding total valuation for those slots. This contradicts the definition of the Demand-Oracle since the utility becomes negative for agent i. □

Since the Demand-Oracle allocates a slot only if that allocation increases the agent's utility, the following result holds.

THEOREM **6.** MAA *is individually rational.*

As the agents' sequence is independent of their reported information, the payment paid by any agent is independent of the information reported by her, and she gets the best of the available slots for her by the Demand-Oracle. Hence, the following result is straightforward.

THEOREM **7.** MAA *is incentive compatible.*

To find the best slot for an agent, the Demand-Oracle 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, the Demand-Oracle requires at most $O(m)$ time for every agent i. Therefore, the following result on the complexity of MAA holds.

THEOREM **8.** MAA *has time complexity $O(mn)$.*

We need a few more results to prove the approximation factor attained by MAA. As few parameters in MAA are different from those in [2], we re-write few results from [2] in a modified form according to MULTI-INDIV-ALLOC. However, we show that the approximation factor remains intact. The restated results (deferred to Appendix D) help us prove the main theorem of this section as follows.

THEOREM **9.** *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* MULTI-INDIV-ALLOC *problem.*

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.

In the following section, we investigate the performance of IMPPreSS in real-world scenarios.

7 Experiments

While IMPPreSS satisfies several desirable properties of a social scheduling mechanism for single slot and divisible multiple slot jobs, its prioritizing profile for different classes of importance, costs of prioritization, reduction in social congestion, and scalability are not theoretically captured. This is why an experimental study is called for.

For these experiments, we consider three discrete levels of valuations of the agents denoted by 3, 2, and 1, which can be interpreted as urgent, medium, and not-urgent 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 multiplying 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 $2\delta^{t-1}$. All optimizations used the Gurobi solver [19] under an academic license.

7.1 Prioritizing profile and its cost

In this section, we investigate what the typical priority slots allotted to an agent of a specific class in IMPPreSSare. The top plot of Figure 1 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.

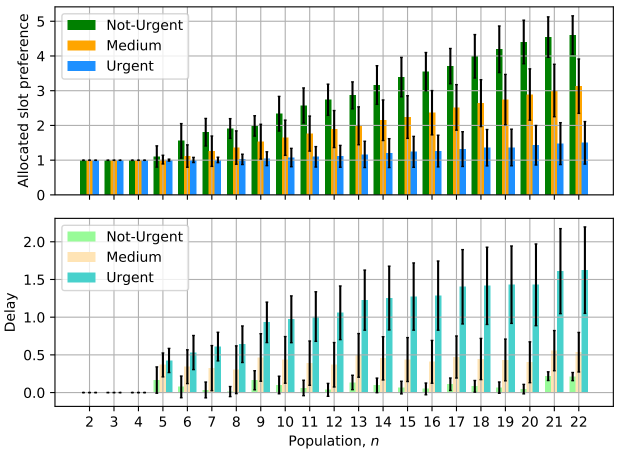

Figure 1: Priority-delay trade-off for IMPPreSS

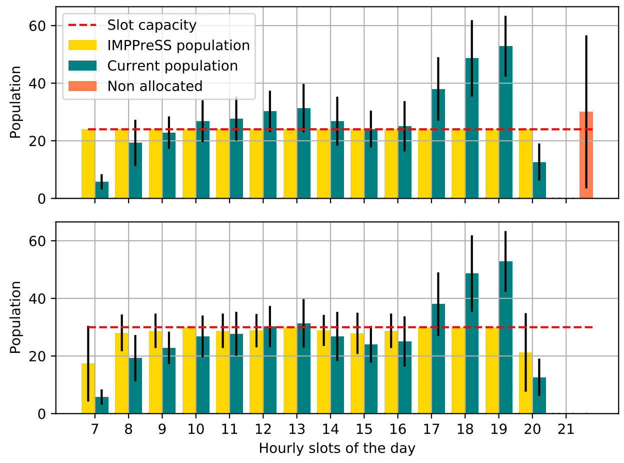

Figure 2: Social congestion reduction, slot capacities 24 (top) and 30 (bottom)

However, IMPPreSS does the prioritized allocations of the agents at the cost of their delays. The bottom plot of Figure 1 shows the corresponding delays decided by IMPPreSS 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.

7.2 Reduction in the social congestion

This section considers real data of customer footfall in a general store that we have collected from the store. The raw dataset is available in the supplementary materials. The dataset contains the customers' 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 of customer identification, we have assumed that the billing timestamps are of unique users for a day. The approximate dimension of the store is 30×30 square-feet. With 6 feet being the minimum distance for a safe social distancing, the recommended number of people in the store at any time should roughly be 16 at any given time. If every shopper takes roughly 30 minutes to shop (including billing and checkout), then in an hour, the safety limit is 32 people an hour. However, the raw data shows that the average hourly population of the store was much above this, particularly during the rush hours (5-8 PM) and on weekends – 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. However, the monthly average of the population in an hour is 26.5, which is very much within the safety limits if social scheduling was implemented. Therefore, this dataset works as a perfect example where users can benefit greatly via social scheduling.

We divide the store opening hours into 14 hourly slots between 7 AM to 9 PM. Under IMPPreSS, the schedules are decided for a day. 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., not-urgent 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 2 shows the comparison of the average current population with that allocated by IMPPreSS for slot capacities of 24 (above) and 30 (below). The figures also show the non-allocated population in red. The figures show the trade-off between better social distancing and its cost (e.g., home delivery). However, in both these cases, the social congestion is reduced by approximately 50% during the rush hours.

7.3 Scalability

This section examines IMPPreSS' computation time for finding the allocation and delays for a realistic population. We run IMPPreSS 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 3 shows the growth of the computation time of the mechanism. As a reference, to solve the allocation and delays for the store of Section 7.2, 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.

7.4 Suboptimality and Running Time Reduction Trade-off for MAA

We presented the mechanism MAA (Algorithm 2) for indivisible multi-slot jobs and proved it to be approximately optimal with a $O(mn)$ time complexity. However, the bound is obtained for a worst-case scenario. In this section, 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 Figure 4 shows the percentage reduction $((t_{\texttt{OPT}} - t_{\texttt{MAA}})/t_{\texttt{OPT}})$ in the running time of MAA compared to the optimal mechanism (which does an exhaustive search as no better algorithm is known for that class of problems), where $t_{\texttt{OPT}}$ and $t_{\texttt{MAA}}$ are the running times of the optimal and MAA mechanisms respectively. The figure's 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

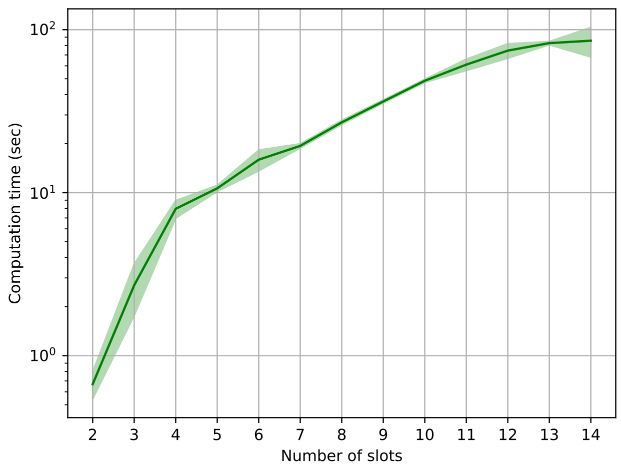

Figure 3: Computation times of `IMPPreSS`

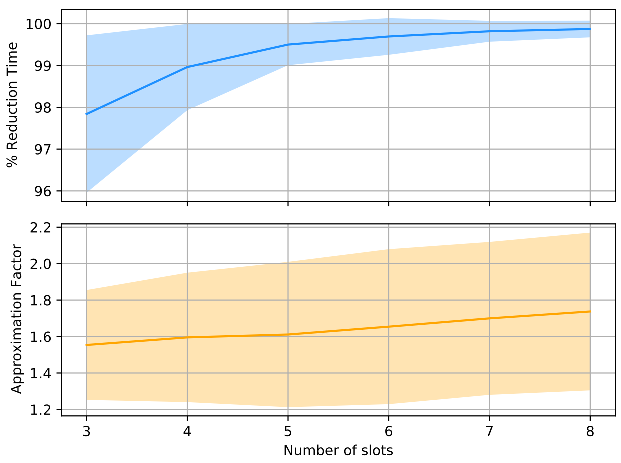

Figure 4: Running time and approximation factor trade-off for `MAA`

insightful result. We see that `MAA` reduces the running time by more than 99.5% and yields an approximation of roughly 1.7 on an average.

8 Summary and Plans of Extension

This paper provides a solution to social distancing using social scheduling and proposes a mechanism that satisfies many desirable theoretical properties for single and divisible multi-slot problems. We show that the indivisible multi-slot problem is NP-complete and provide a poly-time approximately efficient mechanism which satisfies the other design desiderata. Albeit motivated by the recent pandemic, the setup and its solution apply to a more general setting with capacitated slot allocation problems. Experiments show that it prioritizes users based on their importance (at the cost of a cooling-off time). It also reduces the rush-time congestion by a significant proportion without dropping the customers. We have already developed an app that runs `IMPPreSS`. Handling multiple stores simultaneously is a challenging open problem that we want to address in the future.

References

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

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

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

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

[5] Börgers, T.: An introduction to the theory of mechanism design. Oxford University Press, USA (2015)

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

[7] Cavallo, R.: Social distancing equilibrium (2021), http://www.eecs.harvard.edu/~cavallo/wp/cavallo-socdisteq.pdf

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

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

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

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

[12] Friedman, E.J., Parkes, D.C.: Pricing wifi at starbucks: issues in online mechanism design. In: Proceedings of the 4th ACM conference on Electronic commerce. pp. 240–241 (2003)

[13] Ghosh, S., Long, Y., Mitra, M.: Prior-free online mechanisms for queueing with arrivals. Economic Theory pp. 1–30 (2020)

[14] Gibbard, A.: Manipulation of voting schemes. Econometrica **41**, 587–602 (1973)

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

[16] Gosak, M., Kraemer, M.U., Nax, H.H., Perc, M., Pradelski, B.S.: Endogenous social distancing and its underappreciated impact on the epidemic curve. Scientific reports **11**(1), 1–10 (2021)

[17] Grötschel, M., Lovász, L., Schrijver, A.: Complexity, oracles, and numerical computation. In: Geometric Algorithms and Combinatorial Optimization, pp. 21–45. Springer (1993)

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

[19] Gurobi Optimization, L.: Gurobi optimizer reference manual (2020), http://www.gurobi.com

[20] Hajiaghayi, M.T.: Online auctions with re-usable goods. In: Proceedings of the 6th ACM conference on Electronic commerce. pp. 165–174 (2005)

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

[22] Lavi, R., Nisan, N.: Competitive analysis of incentive compatible online auctions. Theoretical Computer Science pp. 310:159–180 (2004)

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

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

[25] Pejó, B., Biczók, G.: 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. pp. 24–31 (2020)

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

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

[28] Roycroft, M., Bhala, N.B., Brockbank, S., O'Donoghue, D., Goddard, A., Verma, A.M.: Rostering in a pandemic: Sustainability is key. Future Healthc J (2020)

[29] Satterthwaite, M.: Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. Journal of Economic Theory **10**, 187–217 (1975)

[30] Schrijver, A.: Combinatorial optimization: polyhedra and efficiency, vol. 24. Springer Science & Business Media (2003)

[31] Shadmehr, M., de Mesquita, E.B.: 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)

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

[33] Thunström, L., Newbold, S.C., Finnoff, D., Ashworth, M., Shogren, J.F.: The benefits and costs of using social distancing to flatten the curve for covid-19. Journal of Benefit-Cost Analysis pp. 1–27 (2020)

[34] Toxvaerd, F.M.: Equilibrium social distancing (2020)

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

[36] Wilder-Smith, A., Freedman, D.O.: 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)

Appendix

A Proof of Proposition 1

Proof. This proof is a standard exercise in the line of the proof for Vickery-Clarke-Groves (VCG) mechanism [10, 18, 35].

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, `IMPPreSS` is dominant strategy truthful in every period. □

B Proof of Proposition 2

Proof. Consider agent i. The utility of i under `IMPPreSS` is $v_i(A^*(v)) - d_i(v)$

$$\begin{aligned}
&= v_i(A^*) - \Big(\sum_{\ell \in N\setminus\{i\}} v_\ell(A^*_{-i}) - \sum_{\ell \in N\setminus\{i\}} v_\ell(A^*) \Big) \\
&= \Big(\sum_{\ell \in N} v_\ell(A^*) - \sum_{\ell \in N} v_\ell(A^*_{-i}) \Big) + v_i(A^*_{-i}) \geqslant 0
\end{aligned}$$

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. □

C Proof of Theorem 5

We denote the decision version of `MULTI-INDIV-ALLOC` with the same name, which is defined as follows:

> Given (N, M, V, L, k, β), is there an allocation $\mathcal{A}$ such that $\sum_{i \in N} \sum_{p \in M,\ j \leqslant p + l_i - 1} \mathfrak{a}_{ip} \leqslant k\ \forall j \in M,\ \sum_{j \in M} \mathfrak{a}_{ij} \leqslant 1\ \forall i \in N$, and $\sum_{i \in N} \sum_{j \in M} v_{ij}\ \mathfrak{a}_{ij} \geqslant \beta$ where $\beta \in \mathbb{R}$.

Proof. To prove that `MULTI-INDIV-ALLOC` is in NP, we have to show that, "given an allocation $\mathcal{A}$, one can verify in polynomial time that $\mathcal{A}$ is a valid solution for `MULTI-INDIV-ALLOC`." The verification will need to solve and check the $(m+n+mn)$ linear constraints and the objective function of Equation (6). Therefore, the computation of the value of objective function and comparison with β is doable in $O(mn)$ time. Hence, `MULTI-INDIV-ALLOC` is in NP.

Next, we show that "every problem in NP is reducible to `MULTI-INDIV-ALLOC` in polynomial time". To prove this, we take a sub-problem of `MULTI-INDIV-ALLOC` and in Lemma 4, we prove that the sub-problem is NP-complete, which implies "every problem in NP is reducible to a sub-problem of `MULTI-INDIV-ALLOC` in polynomial time."

Lemma **4.** *There exist an NP-complete sub-problem of* `MULTI-INDIV-ALLOC`.

Proof. Consider the subset of `MULTI-INDIV-ALLOC` such that the valuation of every agent (v_i) is independent of the starting point of her job during the period. We call this sub-problem as SUB_MULTI. In this proof, we represent the valuation as a tuple $\mathcal{V}$ of length n. Let for SUB_MULTI $k=1$. Mathematically, SUB_MULTI is given by the following integer program (IP).

$$\begin{gathered}\operatorname*{argmax}_{A} \sum_{i\in N}\sum_{j\in M} v_i\ \mathfrak{a}_{ij} \\ \text{s.t.} \sum_{i\in N}\sum_{\substack{p\in M\\ j\leqslant p+l_i-1}} \mathfrak{a}_{ip} \leqslant 1, \forall j\in M, \quad \sum_{j\in M}\mathfrak{a}_{ij}\leqslant 1, \forall i\in N, \quad \mathfrak{a}_{ij}\in\{0,1\}, \forall i\in N, \forall j\in M\end{gathered} \tag{8}$$

We denote the decision version of SUB_MULTI with the same name, which is defined as follows:

> Given $(N, M, \mathcal{V}, L, k, \beta)$, is there an allocation X such that $\sum_{i\in N}\sum_{p\in M,\ j\leqslant p+l_i-1}\mathfrak{a}_{ip}\leqslant 1\ \forall j\in M$, $\sum_{j\in M}\mathfrak{a}_{ij}\leqslant 1\ \forall i\in N$, $\mathfrak{a}_{ij}\in\{0,1\}$, and $\sum_{i\in N}\sum_{j\in M} v_i\ \mathfrak{a}_{ij}\geqslant\beta$ where $\beta\in\mathbb{R}$.

Proof for NP-completeness: It is easy to see that, given an allocation, the verification will need to solve and check the $(m+n+mn)$ linear constraints, and solve the objective function. Therefore, the computation of the value of objective function and comparison with β is doable in polynomial $(O(mn))$ time. Hence, SUB_MULTI is in NP.

Next we have to show that, "every problem in NP is reducible to SUB_MULTI in polynomial time." To prove this, we select the classical KNAPSACK PROBLEM, which is NP-complete and show that it is reducible to SUB_MULTI. The KNAPSACK PROBLEM is as follows: given a set of items $I=\{1,2,\ldots,n\}$, their weights $W=(w_1,w_2,\ldots,w_n)$, profits $P=(p_1,p_2,\ldots,p_n)$ and a knapsack of capacity κ, find a subset Y of items from I such that the total profit of the selected items is maximum, and the total weights of the selected items does not exceed κ. Let y be a vector such that, $y_i=1$ if $i\in Y$, and $y_i=0$ otherwise. The decision version of the knapsack problem is: "Given (I,P,W,κ,β'), is there a vector y such that $\sum_{i\in I} y_i\ w_i\leqslant\kappa$, and for which $\sum_{i\in I} p_i y_i\geq\beta'$ where $\beta'\in\mathbb{R}$."

Consider the following reduction: let the set of items I to be the set of agents N, the items' profits P to be the value of agents' jobs in $\mathcal{V}$, the items' weights W to be the length of the jobs in L, and the capacity of the knapsack κ to be the number of slots m. The KNAPSACK PROBLEM(I,P,W,κ,β') is then reduced (in polynomial time) to SUB_MULTI$(I,\{1,2,\ldots,\kappa\},P,W,\beta')$. And the solution y of KNAPSACK PROBLEM can be constructed from the solution $\mathcal{A}$ of SUB_MULTI such that, $y_i=1$ for $i\in I$, if $\sum_{j\in M}\mathfrak{a}_i=1$ for $i\in N$, else $y_i=0$. □

As `MULTI-INDIV-ALLOC` is verifiable in polynomial time, from Lemma 4 we conclude that `MULTI-INDIV-ALLOC` is NP-complete. □

D Restated results from Bartal et al. [2]

In the following, we will restate a few results from [2] at appropriate places, which will help us prove the main theorem of this section. The lemma and section numbers of these results in the original paper are mentioned within parentheses in the restated lemmata.

Lemma **5** ([2, Section 4.2, Lemma 4])**.** *For every agent i, $v_i(\mathfrak{a}_i^*) \geqslant v_i(\mathfrak{a}_i') - \sum_{j\in[s,s+l_i-1] \text{ s.t. } \mathfrak{a}_{is}'=1} P_j^*$ for every allocation $\mathfrak{a}_i'$, where, P^* is the vector of prices of slots at the end of Algorithm 2.*[12]

From the above result, we get that $\forall i \in N$ and any corresponding customer's allocation $\mathfrak{a}_i'$, $v_i(\mathfrak{a}_i^*) \geqslant v_i(\mathfrak{a}_i') - \sum_{j\in[s,s+l_i-1] \text{ s.t. } \mathfrak{a}_{is}'=1} P_j^*$, where P^* is the vector of prices of slots at the end of Algorithm 2. Let $V(ALG)$ and $V(OPT)$ denotes the sum of valuations of customers for the allocation $\mathcal{A}^*$ given by `MAA`, and that for the optimal allocation (say $\hat{\mathcal{A}}$) respectively. Similarly, $V(ALG^{-b})$ and $V(OPT^{-b})$ represents the sum of valuations of every agent except b according to $\mathcal{A}^*$ and $\hat{\mathcal{A}}$ respectively. Summing it for all the agent $i \in N$, we get the following corollary.

Corollary **2.** $V(ALG^{-b}) \geqslant V(OPT^{-b}) - (k-1)\sum\limits_{j\in M} P_j^*$

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

Lemma **6** ([2, Section 4.2, Lemma 4])**.** $V(ALG^{-b}) \geqslant \frac{\sum_{j\in M} P_j^* - mP^0}{r-1}$

Combining Lemma 6 and Corollary 2, we state the following result.

Lemma **7.** *If $m(k-1)P^0 \geqslant \frac{V(OPT^{-b})}{2}$, then $2((k-1)(r-1)+1) \geqslant V(OPT^{-b})/V(ALG^{-b})$.*

Following the conditions in Lemma 3 and Lemma 7, we fix $P_j^0 = \frac{v_{\max}}{6m(k-1)}, \forall j \in M$, $r = (6m(k-1))^{\frac{1}{k-2}}$. We restate the following result about the approximation ratio from [2].

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

Finally, combining Lemmas 7 and 8, we state the Theorem 9.

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