Time-sharing scheduling with tolerance capacities

George Karakostas[a,1], Stavros G. Kolliopoulos[b,*]

[a]*Department of Computing and Software, McMaster University, 1280 Main St. W., Hamilton, ON, L8S 4K1, Canada*
[b]*Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Panepistimiopolis Ilissia, Athens, 157 84, Greece*

Abstract

Motivated by time-sharing systems with deadlines, such as 2-way synchronization of Digital Twins, we introduce the study of a natural problem which can be abstracted as follows. We are given m machines and n jobs, as well as a set of *tolerance capacities* $u_{ij} \geq 0$ for every job j and machine i. Can we assign the jobs so that, if job j ends up on machine i, the total size of jobs that are processed on i is at most u_{ij}? We define two natural optimization versions: (i) Maximize the total weight of jobs that can be assigned without violating the tolerance capacities u_{ij}, and (ii) minimize the amount $\rho \geq 1$ by which capacities have to be scaled so that all jobs can be assigned. For problem (i), we provide a randomized algorithm with expected approximation factor of $(1 - 1/e - \varepsilon)$, even in the presence of a bound on the number of machines to be used, bundles of required jobs, and required machines, and a randomized algorithm with expected approximation factor of $((1-1/e)^2 - \varepsilon)$ in the presence of machine copies, a bound on the number of machines to be used, and required machines, for any constant $\varepsilon > 0$. For problem (ii), we show that it is $n^{1/2-\varepsilon}$-inapproximable and provide linear integrality gap lower bounds for two key relaxations.

Keywords: time-sharing, tolerance capacities, scheduling

1. Introduction

Time-sharing of resources, such as CPU cycles or network bandwidth, by tasks with timing constraints is a feature of scheduling multitasking systems since their inception in the 1950s [26]. Real-time tasks have strict deadline constraints, and as a result they cannot *tolerate* sharing a time-shared resource with more than a specific number of other tasks. In this work we study the scheduling problem arising from the application of time-sharing.

⋆An extended abstract of this work was presented at *ALGOSENSORS'22* under the title 'Resource time-sharing for IoT applications with deadlines'.

*Corresponding author

Email addresses: karakos@mcmaster.ca (George Karakostas), sgk@di.uoa.gr (Stavros G. Kolliopoulos)

[1]Research supported by an NSERC Discovery grant.

In order to highlight the challenges created by time-sharing scheduling, we describe in detail a recent example of this paradigm, i.e., its application to Digital Twins within the framework of the Internet-of-Things (IoT). An ever increasing number of IoT applications depend on *real-time* response times, i.e., the ability of delivering data to and from the application, as well as processing data, with acceptable delays [29]. These delay requirements can apply to almost all different components of an IoT architecture. For example, the need for fast delivery of the 'freshest' available sensor data to cyber-physical systems has lead to the recent concept of the Age-of-Information [30], i.e., the scheduling of data transmission so that the largest latency between data generation at a source and its delivery to the application is minimized. A different approach to the requirement for timely data delivery and processing, is the imposition of delay constraints (as opposed to delay as objective in AoI), that can guarantee the real-time nature of the system. An example of this latter approach is the concept of *Digital Twins (DT)* [20, 25]. These are virtual replicas of physical systems (PS), which capture a subset of the PS's features, maintain a 2-way synchronization of DT and PS states, and can store data relevant to the PS in order to perform computationally-heavy tasks, such as prediction and data analytics.

The 2-way synchronization requires that a DT performs periodically its data transmission and task processing, and is expected to finish both within a given period, thus assuring the data 'freshness'. With the proliferation of DTs, the time-sharing of critical resources, such as wireless channels and CPUs, by many DTs simultaneously puts a strain on the satisfaction of these timing requirements, and motivates the scheduling problem we introduce in this work as follows: A DT j with a synchronization period T_j, needs r_j CPU cycles in order to complete its data processing task (for simplicity we assume the data transmission time is negligible). DT j's task is executed on a server i of CPU frequency f_i together with the tasks of K other DTs, which share the CPU equally and in a round-robin fashion with j. Each job receives an equal share at each point in time whether it is active or not. In order for j's task to finish on time, the inequality $\frac{Kr_j}{f_i} \leq T_j$ must hold (j gets every K-th cycle of the CPU), which implies that $K \leq \frac{f_i T_j}{r_j}$, i.e., DT j's task can co-execute with at most $u_{ij} := \lfloor \frac{f_i T_j}{r_j} \rfloor$ (including itself) DTs on server i. An example is shown in Figure 1. When DT 1 has the CPU i to itself ($K = 1$ in case (a)), or when it shares CPU cycles only with DT 2 in a round-robin fashion ($K = 2$ in case (b)), DT 1's job can be completed within its synchronization period T_1. But when three DTs share equally the CPU cycles ($K = 3$ in case (c)), the DT 1 job cannot be completed in a single period T_1, and, therefore, $K = 3$ is infeasible. Hence, for this machine, $u_{i1} = 2$, i.e., DT 1 can *tolerate* time-sharing with one other DT, but not two. A similar situation arises when several DTs share a wireless channel with their PSs, using TDMA (Time Division Multiple Access) [17], or, in general, when jobs with completion deadlines have to time-share a common resource. As in Figure 1, a job j deadline results in an upper bound u_{ij} on the number of other jobs that j can tolerate on resource i. Therefore, the natural question of how to schedule a set of tasks with deadlines on a set of resources/machines, when round-robin time-sharing is applied, so that all deadlines are respected, reduces to a scheduling problem of jobs on machines, when every job has an upper bound u_{ij} on the number of jobs (including itself) a job j can tolerate on machine i.

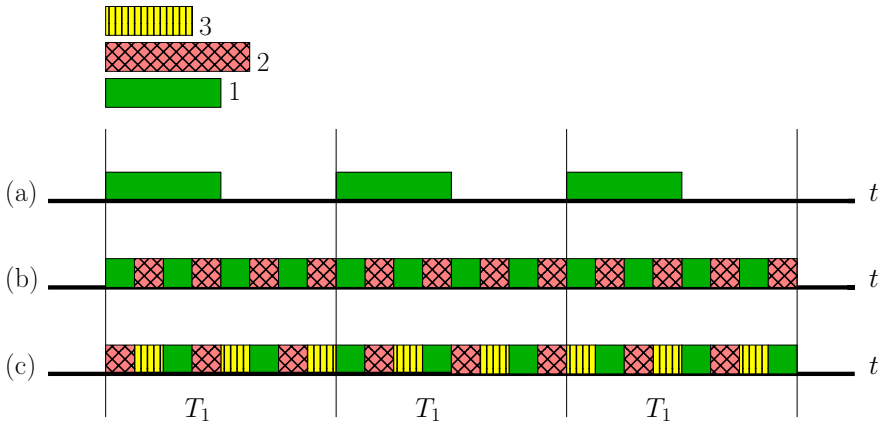


Figure 1: DT 1's job executed without time-sharing (a), round-robin time-sharing with DT 2 (b), and round-robin time-sharing with both DTs 2 and 3 (c). In case (c), DT 1's job cannot be completed within its synchronization period T_1.

This problem can be abstracted in the following way: Given a set of n distinct balls, a set of m distinct bins, and a set of nm nonnegative integer *tolerance capacities* u_{ij}, can the balls be placed in the bins so that if the j-th ball is placed in bin i, at most u_{ij} balls in total are placed in bin i?

The basic problem can be generalized as follows. We have a set of jobs and a set of machines and job j has processing time p_{ij} on machine i. Assigning job j to machine i implies an upper bound u_{ij} on the total *load* (instead of the number of jobs) that job j can tolerate on machine i. Although this is a rather natural generalization of knapsack, we are not aware of any previous work on it. Note that some tolerance capacities u_{ij} can be equal to 0, i.e., job j cannot be assigned to machine i. Formally, the search problem we examine in this paper is the following.

Machine-Sharing with Tolerance Capacities (MSTC)
Input: Set of n jobs $\mathcal{J}$, set of m machines $\mathcal{M}$, and for each $(i,j) \in \mathcal{M} \times \mathcal{J}$, tolerance capacity $u_{ij} \in \mathbb{Z}_{\geq 0}$ and processing time $p_{ij} \in \mathbb{Z}_{\geq 0}$.
Output: An assignment $\sigma : \mathcal{J} \to \mathcal{M}$ such that $\sum_{k \in \sigma^{-1}(\sigma(j))} p_{\sigma(j)k} \leq u_{\sigma(j)j}$ for all $j \in \mathcal{J}$, or NO if no such assignment exists.

For job $j \in \mathcal{J}$, $\mathcal{M}(j)$ denotes the set of machines on which j can be assigned, i.e., $\mathcal{M}(j) = \{i \in \mathcal{M} \mid u_{ij} > 0\}$. Similarly, for $i \in \mathcal{M}$, $\mathcal{J}(i) = \{j \in \mathcal{J} \mid u_{ij} > 0\}$. With this notation in place, MSTC can be equivalently formulated as finding a feasible solution to the following quadratic program:

$$\sum_{i \in \mathcal{M}(j)} x_{ij} \geq 1 \qquad \forall j \in \mathcal{J} \tag{QP}$$

$$x_{ij} \cdot \sum_{k \in \mathcal{J}(i)} p_{ik} x_{ik} \leq u_{ij} \quad \forall j \in \mathcal{J}, \forall i \in \mathcal{M}(j) \tag{1}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in \mathcal{M}, \forall j \in \mathcal{J}(i) \tag{2}$$

The decision version of MSTC can be easily seen to be NP-complete (e.g., via a reduction from SAT). To the best of our knowledge, this is the first time that the problem has been studied. Assignment problems with forbidden *pairs* of assignments have been studied in the literature (e.g., [10]), but they are incomparable to MSTC.

Problem MSTC gives rise to two natural objective functions and the corresponding optimization versions. In the maximization setting, every job j has a weight $w_j \geq 0$. One asks for a *maximum-weight* set of jobs that can be assigned to machines without violating any tolerance capacities. Two well-known optimization problems can be obtained as special cases of our problem. In our terminology the Multiple Knapsack problem corresponds to the case where every job has the same processing time p_j on all the machines and the tolerance capacities of capacities u_{ij} of jobs on machine i are equal to a job-independent value u_i. Multiple Knapsack is known to have a PTAS [3]. Our problem captures further as a special case the Generalized Assignment Problem with machine-independent profits. The latter problem results from Multiple Knapsack if we allow machine-dependent processing times p_{ij} for each job j. The best known approximation ratio for the Generalized Assignment Problem is $(1 - 1/e + \varepsilon)$, for a small absolute constant $\varepsilon > 0$ [9].

The basic maximization formulation of MSTC can be augmented with various additional constraints. An immediate additional constraint is to require that no more than k machines can be used, but further natural constraints include resource augmentation (e.g., more UAVs used as relays at a location to increase the number of available channels [21]), or bundles of jobs that *have* to be executed on some machine, or combinations of the above. We call this general family of problems the Maximum Machine-Sharing with Tolerance Capacities (MMSTC), and we will denote by MMSTC($L_1, L_2, \ldots$) the MMSTC problem with additional constraints labeled $L_1, L_2, \ldots$. In this work we will address the following additional constraints (their label is given in parenthesis):

k-Coverage (k-C): No more than k machines can be used. This is a typical coverage constraint (see, for example, [4]).

Bundles of required jobs (RJ): Given a set C of l disjoint subsets of jobs $C_1, \ldots, C_l$, each bundle C_i *must* be scheduled on a machine executing exclusively this bundle. For example, in a time-sharing (i.e., multitasking) Operating System, there may be background processes that *must* be executed periodically, reserving a machine, regardless of the existence of other tasks.

Required machines (RM): Given a set of machines $A \subseteq \mathcal{M}$, the machines in A *must* be used. For example, if there is pricing differentiation between CPUs used in a server farm, it may be imperative for jobs to use the cheaper CPUs in A before being forced to use the rest, even if this would result in slower execution (i.e., larger p_{ij}'s).

Resource augmentation (A): Given integers $l_1, \ldots, l_{|\mathcal{M}|}$, the schedule can use at most l_i copies of the i-th machine. Note that since there are n jobs, no more than n copies are needed for each machine, i.e., $\sum_{i=1}^{|\mathcal{M}|} l_i \leq mn$.

For example, MMSTC(k-C, RJ) denotes the MMSTC problem where at most k machines can be used, and there are bundles of required jobs (note that there is no resource augmentation or required machines constraint). We denote by MMSTC() the basic maximization problem where we seek to maximize the weight of the assigned jobs without any of the additional constraints k-C, RJ, RM, A. Table 1 summarizes the constraint abbreviations used throughout this paper.

Table 1: Constraint abbreviations used in the paper.

Abbreviation	Constraint
k-C	k-Coverage (use at most k machines)
RJ	Bundles of required jobs
RM	Required machines (must use these machines)
A	Resource augmentation (can use multiple copies of machines)

The minimization problem derived from MSTC is a *minimum-congestion* version that asks for the smallest scaling factor that one can multiply all tolerance capacities with, so that there is a feasible assignment for all jobs.

Definition 1. *For $\rho \geq 1$, an assignment σ of the jobs in $J \subseteq \mathcal{J}$ is* ρ-feasible *if for all $j \in J$, $\sum_{k \in \sigma^{-1}(\sigma(j))} p_{\sigma(j)k} \leq \rho \cdot u_{\sigma(j)j}$.*

In the Scaled Machine-Sharing with Tolerance Capacities (SMSTC) problem we seek a ρ-feasible assignment of $\mathcal{J}$ with minimum ρ. Similarly to MMSTC(), SMSTC generalizes in its turn a classic problem. In particular, the special case of SMSTC where the tolerance capacities u_{ij} are equal to a common value T_i, for all $j \in \mathcal{J}(i)$, is the famous unrelated machine scheduling problem with deadlines for which Lenstra et al. gave a 2-approximation [19]. Our problem is more general, as every job has its own upper bound on the completion time of machine i, namely u_{ij}. It also turns out that our problem is much harder to approximate. In the negative results we outline below for SMSTC, every job j has $p_{ij} = 1$ for every $i \in \mathcal{M}(j)$.

MMSTC() can be efficiently reduced to the Separable Assignment Problem (SAP) of [11] with an implicit set system that describes feasible packings of jobs. The algorithm of [11] applies standard randomized rounding on the Configuration LP relaxation of Section 3.1 and yields an $(1 - \frac{1}{e} - \varepsilon)$-approximation for any constant $\varepsilon > 0$, but does not seem able to handle the additional constraints mentioned above. In Section 3.2 we also design an $(1 - \frac{1}{e} - \varepsilon)$-approximation algorithm for MMSTC() (cf. Theorem 2), using the more sophisticated dependent rounding technique of [27, 12] on the Configuration LP. We prove that this approximation ratio is best possible unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log \log n)})$ (Theorem 6 in Section 3.3). Unlike [11], dependent rounding allows us to extend the results of Section 3.2 to include additional constraints in Section 3.4. We note that the k-C constraints can still be $1 - \frac{1}{e} - \varepsilon$-approximated by the algorithms of [2, 5] that generalize [11] to matroidal constraints, but they do not seem to be able to handle constraints RM, RJ, A (or their combinations); there is no obvious expression of these constraints in a matroid setting. Using

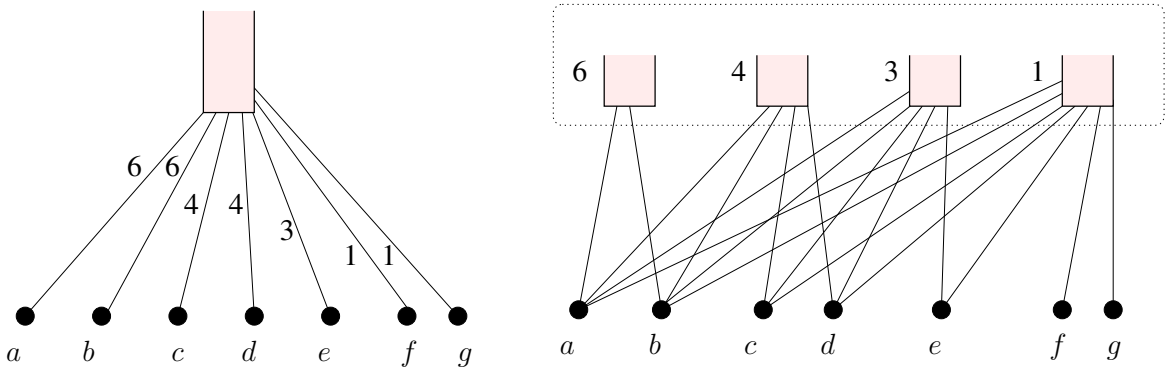
the strong degree properties of dependent rounding, we obtain an approximation factor of $1-\frac{1}{e}-\varepsilon$ for MMSTC(k-C, RJ, RM) (Theorem 8). We also obtain an approximation factor of $(1-\frac{1}{e})^2-\varepsilon$ for MMSTC(k-C, RM, A) (Theorem 9). Clearly, the approximation factors of Theorems 8,9 apply also to the MMSTC problems with a subset of the corresponding constraints. Since we use dependent rounding, the approximation factors in this work are in expectation.

Unfortunately, SMSTC turns out to be much harder to approximate than MMSTC(), even for the case $p_{ij}=1$, for all $i\in\mathcal{M}, j\in\mathcal{J}(i)$. Using a reduction from 3D-MATCHING, we show that there is no polynomial-time $(n^{1/2-\varepsilon})$-approximation algorithm for SMSTC, unless $\mathsf{P}=\mathsf{NP}$ (cf. Theorem 12), where $n=|\mathcal{J}|$. The bound holds even when every job j has the same tolerance capacity $u_{ij}=u_j$ on every machine in $\mathcal{M}(j)$. In order to tackle the problem algorithmically, we explore two key relaxations. First, we study the Configuration LP, a powerful linear relaxation that was introduced in the context of the cutting stock problem [7, 14] and has been used among other for bin packing [16, 23] and scheduling problems with assignment restrictions (e.g., [1, 28, 15]). Applied to the MSTC problem, it is strictly stronger than the natural LP that has assignment variables x_{ij} for job-machine pairs (cf. Proposition 2). We prove that the Configuration LP has an integrality gap of $\Omega(n)$ for congestion even when there are only two distinct tolerance capacity values, every job j has the same tolerance capacity u_j on every machine, and each job can be assigned to at most two machines (cf. Theorem 10). The second relaxation we consider is the formulation resulting from the quadratic program (QP) by relaxing the integrality constraints. Notably, this is a non-convex program. Still we show that it has an integrality gap of at least m, the number of machines. The lower bound holds again when every job has a machine-independent tolerance capacity (cf. Theorem 11). Hence, rounding the fractional solution of these two key formulations cannot give a non-trivial approximation factor.

The outline of the paper is as follows. In Section 2 we provide an equivalent reformulation of the MSTC problem that provides a helpful abstraction. In Section 3 we study the approximability of MMSTC giving both positive and negative results. In Section 4 we prove integrality gap lower bounds and hardness of approximation for SMSTC. We conclude in Section 5 with some open problems.

2. An equivalent formulation of MSTC

In this section we provide an equivalent formulation of the MSTC problem. We define the *Submachine Scheduling Problem (SSP)* as follows. We are given sets $\mathcal{M}$, $\mathcal{J}$, of machines and jobs respectively, and a set of allowed job-machine assignments that defines the sets $\mathcal{J}(i)$, $i\in\mathcal{M}$, and $\mathcal{M}(j)$, $j\in\mathcal{J}$. For every $i\in\mathcal{M}$, there is a finite set S_i of m_i submachines. Submachine $k\in S_i$ has capacity $\bar{u}_{ik}$. Let us number the submachines so that $\bar{u}_{i1}<\bar{u}_{i2}<\ldots<\bar{u}_{im_i}$. Job $j\in\mathcal{J}(i)$ has processing time p_{ij} on machine i and there is an $h(j)\in[m_i]$ so that j can be assigned only to the set of *allowed submachines* $\mathcal{M}(i,j):=\{1,\ldots,h(j)\}$. A *submachine assignment* of the set $J\subseteq\mathcal{J}$ is a mapping $\psi: J\to\cup_{i\in\mathcal{M}}S_i$ such that (i) for all $j\in J$, $\psi(j)\in\mathcal{M}(i,j)$ for some machine i, and (ii) for all $i\in M$, at most one



(a) MSTC instance

(b) SSP instance

Figure 2: (a) An MSTC instance with one machine and seven jobs and the equivalent instance with four submachines. The edges denote the allowed assignments. The numbers on the edges denote the job tolerance capacities u. (b) The equivalent SSP instance. The number next to a submachine denotes the submachine capacity $\bar{u}$. In the SSP instance only one of the submachines can be chosen to process jobs.

of the sets $\psi^{-1}(k)$, $k \in S_i$, is nonempty. In words, every job j in J is assigned to an allowed submachine of a machine in $\mathcal{M}(j)$ and for every machine $i \in \mathcal{M}$, at most one of the submachines in S_i can be "open". The submachine assignment ψ is *ρ-feasible,* for $\rho \geq 1$, if $\sum_{j \in \psi^{-1}(k)} p_{ij} \leq \rho \bar{u}_{ik}$, $\forall i \in \mathcal{M}, \forall k \in S_i$. We note that if we are only interested in 1-feasible assignments, job j can effectively be assigned only to machines $l(j)$ to $h(j)$ where $l(j) = \min\{x \in [m_i] \mid \bar{u}_{ix} \geq p_{ij}\}$. We define a mapping ζ that maps an SSP instance $\mathcal{I}$ to an MSTC instance $\zeta(\mathcal{I})$ with the same sets of machines, jobs and allowed job-machine assignments, where the tolerance capacity of job j on machine i is equal to $\bar{u}_{ih(j)}$.

Conversely, we define a mapping η that maps an MSTC instance $\mathcal{I}$ to an SSP instance $\eta(\mathcal{I})$. Recall that we are given as input a set $\mathcal{M}$ of machines, a set $\mathcal{J}$ of jobs and a set $\{u_{ij}, p_{ij} \in \mathbb{Z}_{\geq 0} \mid (i,j) \in \mathcal{M} \times \mathcal{J}\}$. For $i \in \mathcal{M}$, let $d(i)$ denote $|\mathcal{J}(i)|$. Sort the capacities u_{ij}, $j \in \mathcal{J}(i)$, in non-decreasing order $u_{ij_1} \leq u_{ij_2} \leq \ldots \leq u_{ij_{d(i)}}$. Let m_i denote the number of distinct values in the sequence $u_{ij_1}, u_{ij_2}, \ldots, u_{ij_{d(i)}}$. Denote these distinct values in increasing order as $\bar{u}_{i1} < \bar{u}_{i2} < \ldots < \bar{u}_{im_i}$. We define for each machine $i \in \mathcal{M}$, a set S_i of m_i *submachines* where submachine $k \in S_i$ has capacity $\bar{u}_{ik}$. The set of jobs that are allowed to be assigned to submachine $k \in S_i$ is defined as $\mathcal{J}(i,k) := \{j \in \mathcal{J}(i) \mid u_{ij} \geq \bar{u}_{ik}\}$. If we denote by $\mathcal{M}(i,j)$ the set of submachines of machine i to which job j can be assigned, it follows from the definition of the $\mathcal{J}(i,k)$ sets that $\mathcal{M}(i,j) = [h(j)]$, where $h(j) = \max\{k \in m_i \mid \bar{u}_{ik} \leq u_{ij}\}$.

It is easy to see that the two mappings are invertible and $\eta = \zeta^{-1}$. Given an instance $\mathcal{I}$ of MSTC by a *submachine assignment* of the jobs we mean a submachine assignment for the SSP instance $\eta(\mathcal{I})$. The next proposition makes precise the notion that the two instances $\mathcal{I}$ and $\eta(\mathcal{I})$ are equivalent. See Figure 2 for an example.

Proposition 1. *Given the input $\{u_{ij}, p_{ij} \in \mathbb{Z}_{\geq 0} \mid (i,j) \in \mathcal{M} \times \mathcal{J}\}$, and a set $J \subseteq \mathcal{J}$, there is a ρ-feasible assignment σ of the jobs in J iff there is ρ-feasible submachine assignment ψ*

of J.

Proof. Let σ be a ρ-feasible assignment. For machine i, let l_i be the job in $\sigma^{-1}(i)$ with the smallest tolerance capacity. It follows that $\sum_{j\in\sigma^{-1}(i)} p_{ij} \leq \rho \cdot u_{il_i}$. Value u_{il_i} is equal to $\bar{u}_{ik_i}$ for some $k_i \in S_i$. Consider the submachine assignment ψ where for every $i \in \mathcal{M}$, submachine $k_i \in S_i$ is open, and $\psi^{-1}(k_i) = \sigma^{-1}(i)$. It follows that ψ is a ρ-feasible submachine assignment.

For the converse, if a ρ-feasible submachine assignment ψ is given, let k_i be the submachine of machine i that is open and $\bar{u}_{ik_i}$ its capacity. Define $\sigma : J \to \mathcal{M}$ such that for every i, $\sigma^{-1}(i) = \psi^{-1}(k_i)$. We have that $\sum_{j\in\sigma^{-1}(i)} p_{ij} = \sum_{j\in\psi^{-1}(k_i)} p_{ij} \leq \rho \cdot \bar{u}_{ik_i}$. Moreover since the set of jobs $\psi^{-1}(k_i)$ is a subset of $\mathcal{J}(i,k_i)$ we have that $\bar{u}_{ik_i} \leq u_{ij}$ for each $j \in \psi^{-1}(k_i)$. Thus for each $j \in \sigma^{-1}(i)$, $\sum_{k\in\sigma^{-1}(i)} p_{ik} \leq \rho \cdot u_{ij}$. Therefore σ is ρ-feasible. □

In the rest of the paper we will choose each time the problem formulation (with or without submachines) that is more convenient.

3. Approximation algorithms for MMSTC

We consider the MMSTC family of problems, i.e., given input $\{u_{ij}, p_{ij} \in \mathbb{Z}_{\geq 0} \mid (i,j) \in \mathcal{M}\times\mathcal{J}\}$, and weights w_j for each job $j \in \mathcal{J}$, find a maximum-weight $S \subseteq \mathcal{J}$ for which there is a 1-feasible assignment that also respects a subset of constraints k-C, RJ, RM, A.

The basic problem MMSTC() is an instance of the general Separable Assignment Problem (SAP) defined in [11], and, therefore, the standard randomized rounding of the Configuration LP of Section 3.1, as analyzed in [11], implies a $(1-\frac{1}{e})$-approximation algorithm. Unfortunately, this simple rounding does not seem able to handle the extra constraints defined above. We will employ the powerful machinery of *dependent rounding* described in Section 3.2, to approximate the more constrained versions of MMSTC in Section 3.4.

3.1. Linear relaxation with configurations

For machine i and submachine $k \in S_i$, a subset $C \subseteq \mathcal{J}(i,k)$ is a *configuration* if $\sum_{j\in C} p_{ij} \leq \bar{u}_{ik}$. The set of these configurations is denoted $\mathcal{C}(i,k)$. The Configuration LP, denoted (CLP), has a variable x_{i,C_k} for each machine i, submachine $k \in S_i$, and configuration $C_k \in \mathcal{C}(i,k)$:

$$\max \sum_{j\in\mathcal{J}} w_j \left(\sum_{i\in\mathcal{M}} \sum_{k\in S_i} \sum_{C_k:\, j\in C_k} x_{i,C_k} \right) \text{ s.t.} \tag{CLP}$$

$$\sum_{k\in S_i} \sum_{C_k} x_{i,C_k} = 1 \qquad \forall i \in \mathcal{M} \tag{3}$$

$$\sum_{i\in\mathcal{M}} \sum_{k\in S_i} \sum_{C_k: j\in C_k} x_{i,C_k} \leq 1 \qquad \forall j \in \mathcal{J} \tag{4}$$

$$x_{i,C_k} \geq 0 \qquad \forall i \in \mathcal{M}, \forall k \in S_i, \forall C_k \in \mathcal{C}(i,k) \tag{5}$$

The set of constraints (3) ensures that each machine is assigned one configuration and that one submachine is open. Constraints (4) ensure that each job is assigned at most once. Clearly, an integer solution to (CLP) corresponds to a 1-feasible assignment for a maximum-weight subset of $\mathcal{J}$. For a configuration C_k, let $w(C_k) := \sum_{j \in C_k} w_j$. The dual of (CLP) is the following:

$$\min \sum_{i \in M} y_i + \sum_{j \in \mathcal{J}} z_j \quad \text{s.t.} \tag{D-CLP}$$

$$y_i + \sum_{j \in C_k} z_j \geq w(C_k) \qquad \forall i \in \mathcal{M}, \forall k \in S_i, \forall C_k \in \mathcal{C}(i,k) \tag{6}$$

$$z \geq 0 \tag{7}$$

$$y \text{ free} \tag{8}$$

Since there is an exponential number of configurations, (CLP) has an exponential number of variables, and, therefore, it cannot be solved directly in polynomial time. Instead, its dual (D-CLP) will be solved using an approximate separation oracle. In what follows, we will use the notion of a *β-approximate separation oracle* [11], i.e., an algorithm that given values y, z either returns a violated constraint or guarantees that values $y/\beta, z$ satisfy constraints (6)-(8). The running time of the separation oracle depends on the number of variables of (D-CLP).

Lemma 1. *There is a polynomial-time $(1-\varepsilon)$-approximate separation oracle for* (D-CLP)*, for any constant $\varepsilon > 0$.*

Proof. Given a candidate solution (y, z) to (D-CLP), its separation oracle has to solve $\sum_{i \in \mathcal{M}} |S_i|$ instances of a Knapsack problem, one for each $i \in \mathcal{M}$ and $k \in S_i$, defined as follows. Let the set of items be $\mathcal{J}(i,k)$ and the knapsack capacity $\bar{u}_{ik}$. Every item j has a size p_{ij}. We remove from $\mathcal{J}(i,k)$ all items j with $p_{ij} > \bar{u}_{ik}$. The objective of the Knapsack problem will be to pack a subset of items of maximum value. We proceed to define the values of the items.

Every item $j \in \mathcal{J}(i,k)$ has a (possibly negative) value $v_j = w_j - z_j$. Let $J' = \{j \in \mathcal{J}(i,k) \mid v_j \geq 0\}$. The addition of items with negative values can only decrease the objective function of the Knapsack problem.

If $J' = \varnothing$, the maximum-value solution for our Knapsack consists of a single item, the one with value equal to $\max_{j \in \mathcal{J}(i,k)} v_j$. If $J' \neq \varnothing$ only items in J' will be considered for inclusion in the knapsack. We run the standard Knapsack FPTAS on input J'. It is well-known that by rounding all nonnegative $v_j \geq 0$ down to $\lfloor \frac{nv_j}{\varepsilon v_{max}} \rfloor$, the Dynamic Programming algorithm solving knapsack runs in polynomial time and will detect whether there is a feasible packing with total value that exceeds $(1-\varepsilon)V_{max}$, where $V_{max} = \max\{\sum_{j \in C_k}(w_j - z_j) : C_k \in \mathcal{C}(i,k)\}$. If the value of the solution returned by the FPTAS is greater than y_i, then we have detected a violated constraint, otherwise we have $(1-\varepsilon)V_{max} \leq y_i$, i.e., the total value of any configuration for this pair of i, k is no greater than $y_i/(1-\varepsilon)$.

When we can no longer detect any violated constraint for all i, k, then the values $y/(1-\varepsilon), z$ satisfy constraints (6)-(8). Therefore the separation oracle is a polynomial $(1-\varepsilon)$-approximate separation oracle, for any constant $\varepsilon > 0$. □

It is well-known (cf. Lemma 2.2 in [11]) that such a separation oracle implies a polynomial-time $(1-\delta)$-approximate algorithm for solving (CLP), for any constant $\delta > 0$ (without any constraint violations). In the produced (CLP) solution a polynomial number of variables will be set to a nonzero value, in particular the variables that correspond to the dual constraints checked by the separation oracle. This approximate (fractional) solution of (CLP) will be rounded in the next section.

3.2. Dependent-rounding algorithm

Let x^* be an $(1-\delta)$-approximate (fractional) solution of (CLP), computed as described in Section 3.1. The vector x^* induces on each machine a probability distribution on the submachines in S_i. We will use dependent rounding to choose one configuration per machine and ensure that a near-optimal fraction of jobs is scheduled.

Srinivasan [27] (see also [12]) has provided a technique to sample algorithmically from a distribution with the following properties. Consider any sequence of t reals $P = (p_1, \ldots, p_t)$ such that $p_i \in [0,1]$ and $\sum_i p_i = l$, for an integer l. Srinivasan [27] defines a distribution $D(t;\ P)$ over vectors in $\{0,1\}^t$ such that any vector $(X_1, \ldots, X_t)$ sampled from $D(t; P)$ satisfies the following three properties.

(A1) *(probability preservation)* $\forall i, \Pr[X_i = 1] = p_i$.

(A2) *(degree preservation)* $\Pr[|\{i\colon X_i = 1\}| = l] = 1$.

(A3) *(negative correlation)* For all $S \subseteq [t]$ we have $\Pr[(\bigwedge_{i\in S}(X_i = 0)] \leq \prod_{i\in S} \Pr[X_i = 0]$ and $\Pr[(\bigwedge_{i\in S}(X_i = 1)] \leq \prod_{i\in S} \Pr[X_i = 1]$.

The existence of the distribution is established algorithmically:

Theorem 1 ([27])**.** *Given $P = (p_1, \ldots, p_t)$ there is a linear-time algorithm that generates a sample from distribution $D(t; P)$.*

Let $\mathcal{C}_i$ denote the disjoint union of the sets of configurations in $\bigsqcup_{k\in S_i} \mathcal{C}(i,k)$ whose corresponding variable has a nonzero value in the solution x^* of (CLP). Note that every configuration in $\mathcal{C}_i$ belongs to a unique $\mathcal{C}(i,k)$. Denote by $x^*|i$ the restriction of vector x^* to the entries corresponding to the configurations in $\mathcal{C}_i$. To simplify notation, set $t_i = |\mathcal{C}_i|$. We define a distribution $D(t_i;\ x^*|i)$ that satisfies properties (A1), (A2), (A3) for each machine $i \in \mathcal{M}$. Observe that in our setting $l = 1$. The rounding algorithm is the following.

ALGORITHM DEPROUND
For all $i \in \mathcal{M}$, do independently:

1. Using the algorithm of Theorem 1, sample from $D(t_i; x^*|i)$ to obtain vector $X^{(i)} \in \{0,1\}^{t_i}$. By Property (A2), $X^{(i)}$ has a unique entry equal to 1.
2. Assign the configuration C that corresponds to the nonzero entry of $X^{(i)}$ to machine i.

We show the following for MMSTC() (which can also be obtained by standard randomized rounding [11]):

Theorem 2. *Algorithm* DepRound *runs in polynomial time and outputs a* 1*-feasible assignment for a set of jobs* S *whose expected total weight is at least* $(1 - 1/e - \varepsilon)$ *times the optimum of the* (CLP) *relaxation, for any constant* $\varepsilon > 0$*.*

Proof. Let $\mathcal{C}$ denote the disjoint union $\bigsqcup_{i \in \mathcal{M}} \mathcal{C}_i$, i.e., every configuration in $\mathcal{C}$ corresponds to a unique (i, k) pair. For $j \in \mathcal{J}$, let z_j be the random variable that takes value 1 if job j is assigned by Algorithm DepRound and 0 otherwise. The analysis follows the one in [27] for Maximum Coverage versions of Set Cover. We slightly abuse notation and index the entries of the vectors $X^{(i)}$ by the corresponding configurations. Since every configuration C belongs to a unique $\mathcal{C}_i$ we omit the superscript i as well.

$$\begin{aligned}
\Pr[z_j = 1] =& 1 - \Pr[\bigwedge_{C \in \mathcal{C}: C \ni j} (X_C = 0)] \\
\geq& 1 - \prod_{C \in \mathcal{C}: C \ni j} \Pr[X_C = 0] && (9) \\
=& 1 - \prod_{C \in \mathcal{C}: C \ni j} (1 - x^*_C) && (10)
\end{aligned}$$

Inequality (9) follows from the negative correlation property (A3), and equality (10) from property (A1). Define $z^*_j := \sum_{C \in \mathcal{C}: C \ni j} x^*_C$. This is the fractional amount by which job j is scheduled, and the objective value of the solution x^* is equal to $\sum_{j \in \mathcal{J}} w_j z^*_j$. Using the Arithmetic Mean-Geometric Mean inequality and the fact that $z^*_j \leq 1$, it is easy to see that

$$\prod_{C \in \mathcal{C}: C \ni j} (1 - x^*_C) \leq (1 - z^*_j/s)^s$$

where s is the maximum number of configurations in the support of x^* that a job belongs to. By calculus, $1 - (1 - z^*_j/s)^s \geq (1 - (1 - 1/s)^s) \cdot z^*_j > (1 - 1/e) \cdot z^*_j$.

Recall that there is a polynomial-time $(1 - \delta)$-approximate algorithm for solving (CLP), for any constant $\delta > 0$ (without any constraint violations). Therefore, the overall approximation factor is $(1 - \delta)(1 - 1/e) = (1 - 1/e - \varepsilon)$ for $\delta := \varepsilon/(1 - 1/e)$, and the theorem follows. □

In Theorem 6 we show a matching hardness of approximation result for MMSTC().

3.3. Hardness of approximation for MMSTC()

We first establish APX-hardness for the case of unit processing times. In the *Generalized Assignment Problem (GAP)* we are given as input a set B of m bins (knapsacks) and a set S of n items. Each bin $i \in B$ has a capacity $c(i)$ and for each item j and bin i, we are given a size $s(i, j)$ and a profit $p(i, j)$. The objective is to find a subset $U \subseteq S$, that has a feasible packing in B and maximized the profit of the packing. Chekuri and Khanna [3] showed the following negative result for GAP.

Theorem 3. *[3] GAP is* APX*-hard even on instances of the following form for all positive* δ*. (i)* $p(i, j) = 1$ *for all bins* i *and items* j *(ii)* $s(i, j) = 1$ *or* $s(i, j) = 1 + \delta$ *for all bins* i *and items* j *(iii)* $c(i) = 3$ *for all bins* i*.*

Using Theorem 3 we prove APX-hardness for MMSTC() with unit processing times.

Theorem 4. *MMSTC() is* APX*-hard even if (i) for all $i \in \mathcal{M}$ and $j \in \mathcal{J}$, $p_{ij} = 1$, (ii) for all $j \in \mathcal{J}$, $w_j = 1$, and (iii) for all $i \in \mathcal{M}$ and $j \in \mathcal{J}$, $u_{ij} \in \{2, 3\}$.*

Proof. Let $\mathcal{I}$ be a GAP instance with the properties defined in Theorem 3. We will show that every such GAP instance can be reduced in an approximation-preserving manner to an MMSTC() instance $\mathcal{I}'$.

There is a 1-1 correspondence between the bins in the GAP instance and the set of machines $\mathcal{M}$ in the MMSTC() instance. There is a 1-1 correspondence between the set S of items in $\mathcal{I}$ and the set of $\mathcal{J}$ of jobs in $\mathcal{I}'$. All jobs $j \in \mathcal{J}$ have weight $w_j = 1$. For bin i let $J_1^i = \{j \mid s(i,j) = 1\}$ and $J_{1+\delta}^i = \{j \mid s(i,j) = 1 + \delta\}$. All items can be assigned to all bins, albeit with a different size, hence for every machine i, $\mathcal{J}(i) = \mathcal{J}$. We define $p_{ij} = 1$ for all $i \in \mathcal{M}$ and $j \in \mathcal{J}$. Every machine $i \in \mathcal{M}$ has exactly two submachines.

- Submachine 1 has capacity 2 and the set of jobs $\mathcal{J}(i,1)$ that can be assigned to it is $J_1^i \cup J_{1+\delta}^i$.
- Submachine 2 has capacity 3 and the set of jobs $\mathcal{J}(i,2)$ that can be assigned to it is J_1^i.

We show that any feasible assignment σ of items to bins can be mapped to the same assignment σ of jobs to machines where the total profit is equal to the total weight. For bin i there are two cases.
Case 1: $\sigma^{-1}(i) \cap J_{1+\delta}^i = \varnothing$. Then $\sigma^{-1}(i)$ consists of at most 3 items from J_1^i. In the MMTSC() solution we open submachine 2 and assign to it the jobs in $\sigma^{-1}(i)$.
Case 2: $\sigma^{-1}(i) \cap J_{1+\delta}^i \neq \varnothing$. Then $\sigma^{-1}(i)$ consists of at most 2 items from $J_1^i \cup J_{1+\delta}^i$. In the MMTSC solution we open submachine 1 and assign to it the jobs in $\sigma^{-1}(i)$.

Conversely, it is easy to see that any feasible solution to the MMSTC() instance $\mathcal{I}'$ can be mapped to a feasible solution for $\mathcal{I}$ with the same objective function value. Therefore any ρ-appproximate solution for $\mathcal{I}'$ yields a ρ-approximate solution for $\mathcal{I}$. □

To prove an $(1-1/e)$-hardness for general processing times we reduce from the *Distibuted Caching Problem without Bandwidth constraints* (denoted *CapDC*) defined by Fleischer et al. [11]. In the CapDC problem we are given a set U of m cache locations with capacity A_i for each location i and a set H of n requests. There are k request types $\{t_1, \ldots, t_k\}$, $k \leq n$; each request type t_l has a size a_{t_l}, $l \in [k]$. Request j has a request type $t(j)$. The profit of providing request j from cache location i is f_{ij}. A set of requests S_i is feasible for cache location i if it satisfies the capacity constraint: $\sum_{t \in \{t(j) | j \in S_i\}} a_t \leq A_i$. The goal is to find a feasible assignmemt of requests to cache locations to maximize the total profit. Call $(0,1)$-CapDC the special case of CapDC where for all i, j $f_{ij} \in \{0, 1\}$.

For convenience, let us define GraphCapDC as the special case of CapDC where for all i, j, $f_{ij} \in \{0, 1\}$ and a bipartite graph H determines the profit value. If edge $ij \in E(H)$, we say that that request j *can be assigned* to location i and we define profit $f_{ij} = 1$. Else, if j cannot be assigned to location i, $f_{ij} = 0$. Rephrasing a result from [11] in the above

terminology, Fleischer et al. showed that GraphCapDC is hard to approximate even when every cache location can admit at most one request type:

Theorem 5. *[11] There is a family F of instances of GraphCapDC with m cache locations where the number of types and requests are both bounded by $n = \Theta(m^c)$, $c > 0$ a constant, that is not approximable within a factor better than $1 - 1/e + \varepsilon$, for any fixed $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log \log n)})$. The hardness result holds even for GraphCapDC instances where (i) the capacity A_i of each cache location is equal to 1 (ii) the size of each request type is equal to 1.*

We are now ready to show the following.

Theorem 6. *There is a family of instances of MMSTC() that is not approximable within a factor better than $1-1/e+\varepsilon$, for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log \log n)})$. The size of an instance in the family is $n = \Theta(m^c)$, $c > 0$ a constant, where m is the number of machines. The hardness result holds even for MMSTC() instances where for all $i \in \mathcal{M}$, $j \in \mathcal{J}(i)$, (i) $p_{ij} = p_j$ and (ii) $u_{ij} = u_j$, i.e., a job has the same processing time and tolerance capacity on all machines it can be assigned to.*

Proof. We produce an approximation-preserving reduction f from GraphCapDC to MMSTC(). To an instance $\mathcal{I}$ of GraphCapDC we map the instance $f(\mathcal{I})$ which is defined as follows. There is a 1-1 correspondence between the set of cache locations U and the set $\mathcal{M}$ of machines. Similarly, there is a 1-1 correspondence between the set H of n requests and the set $\mathcal{J}$ of jobs. For every job j, $w_j = 1$. We say that job j *can be assigned* to machine i, equivalently $j \in \mathcal{J}(i)$, if request j can be assigned to cache location i. Since a request j has a type t_j, the corresponding job j inherits the type t_j. We need to express in $f(\mathcal{I})$ the constraint that every machine can accept jobs of at most one type. We define the auxiliary sequence of integers $b_1, \ldots, b_n$ where $b_1 = 1$ and for $i \in \{2, \ldots, n\}$, $b_i = nb_{i-1} + 1$.

Let μ_i be the number of types that can be assigned to machine i, i.e., the number of types for which there is at least one job of this type that can be assigned to machine i. Let the corresponding *i-compatible* sequence of types be $(t_{k_1}, \ldots, t_{k_{\mu_i}})$ where $k_1 < \ldots < k_{\mu_i}$. We emphasize that $(k_1, \ldots, k_{\mu_i})$ is a subsequence of $(1, \ldots, k)$. We define the cardinality of the set S_i of submachines of i to be μ_i. Let n_l, $l \in \mu_i$, be the number of jobs of type t_{k_l} that can be assigned to machine i. We now specify to which submachines in S_i each job in $\mathcal{J}(i)$ can be assigned. The n_1 jobs of type t_{k_1} can be assigned only to submachine $1 \in S_i$. Each has processing time $q_1 = b_{k_1}$. The capacity $\bar{u}_{i1}$ of submachine 1 is defined to be nq_1. The n_2 jobs of type t_{k_2} can be assigned to submachines 1 and 2. Each has processing time $q_2 = b_{k_2}$. The capacity $\bar{u}_{i2}$ of submachine 2 is defined to be nq_2. In general, submachine l, $l \in [\mu_i]$, $l \geq 2$, has capacity $\bar{u}_{il} = nq_l$ where all jobs of type t_{k_l} have processing time $q_l = b_{k_l}$ and can be assigned to submachines 1 through l. From the definition of the b-sequence, $q_l > \bar{u}_{ij}$ for every $j < l$. Therefore in any 1-feasible assignment, a job can be assigned to submachine l iff its type is t_{k_l}. From the definition of the instance $f(\mathcal{I})$, a job of type t_k has processing time b_k and tolerance capacity nb_k on all machines in $\mathcal{M}(j)$. This establishes conditions (i) and (ii) of the theorem.

We define a mapping between solutions to $\mathcal{I}$ and solutions to $f(\mathcal{I})$. In any feasible solution of the GraphCapDC instance $\mathcal{I}$, location i hosts requests of at most one type, say t_l, $l \in [k]$. In the instance $f(\mathcal{I})$, let p be such that t_l is the pth type in the i-compatible sequence, i.e., $t_l = t_{k_p}$. In the solution for $f(\mathcal{I})$ we open submachine p and assign to it the jobs corresponding to the requests assigned to location i. Conversely, consider any 1-feasible solution for the MMSTC() instance $f(\mathcal{I})$ and let p be the index of the open submachine for machine i. By the discussion above, jobs of at most one type have been assigned to submachine p, in particular jobs of the type t_{k_p}.

It is straightforward to verify that the solution mappings defined above translate the total profit to the exact same total weight and vice versa. Therefore the reduction f is approximation-preserving. The family F of hard GraphCapDC instances defined in Theorem 5, gives rise to a family $f(F)$ of hard MMSTC() instances with m machines where the number of jobs, and submachines are bounded by a polynomial in m, while the capacities and processing times can be represented each by a number of bits bounded also by a polynomial in m. By Theorem 5, if we can approximate MMSTC() on the family $f(F)$ in polynomial time within a factor less than $1 - 1/e + \varepsilon$, then $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log \log n)})$. □

3.4. MMSTC with additional constraints

The techniques of the previous section can be used to derive good approximations for the extension of MMSTC() with additional constraints. First, we will show that Theorem 2 extends to MMSTC(k-C, RJ, RM). Then, we will show that MMSTC(k-C, RM, A) is $((1-\frac{1}{e})^2-\varepsilon)$-approximable for any constant $\varepsilon > 0$. Note that the introduction of constraints A leads to a worsening of the approximation factor. Also, note that MMSTC with any subset of the constraints introduced in these two problems can be polynomially solved within the same approximation factor.

For both problems, we will use the extension of [27] to bipartite graphs done by [12]. More specifically, the dependent rounding of Section 3.2 becomes the rounding procedure of [12] on a bipartite graph $G = (U, V, E)$, constructed as follows: Each distribution $D(t_i;\ x^*|i)$ gives rise to a star with machine i at its center, and the configurations of $\mathcal{C}_i$ in the support of x^* at the leaves. Side U contains the machines/centers of these stars, while the leaves of each star centered at machine i correspond to the configurations in $\mathcal{C}_i$, and are vertices on side V. In order to model the additional constraints, the following changes will be made to G. Please see Table 1 for the constraint abbreviations.

k-C: We add a "dummy" configuration vertex $C_\emptyset$ to V. The assignment of this "dummy" configuration to a machine will imply that this machine *cannot* be used. Note that $C_\emptyset$ can be assigned to more than one machines.

RJ: Let $\mathcal{C}^R = \{C_1, C_2, \ldots, C_l\}$ be the set of bundles of required jobs. For each one of the l disjoint subsets of jobs $C_1, \ldots, C_l$ that have to be scheduled as a bundle, we create a new vertex in V. The configurations in $\mathcal{C}^R$ do not appear amongst the vertices of $\mathcal{C}_i$ for any machine i.

RM, A: No changes to G are needed.

An edge $(i, C) \in E$ represents the assignment of configuration C to machine i, and corresponds to decision variable $x_{i,C}$. Note that C can be $C_\emptyset$, in which case edge $(i, C_\emptyset)$ indicates leaving machine i unused. There is an edge $(i, C) \in E$ iff $x^*_{i,C} \in (0, 1)$ in the fractional solution x^* we are rounding. Note that there are no edges for initially integral (0 or 1) $x^*_{i,C}$, and edge (i, C) disappears when $x^*_{i,C}$ is rounded to 0 or 1 during the rounding process.

Problem MMSTC(k-C, RJ, RM). To simplify our exposition, we will assume that we have guessed the *exact* number of machines $k_0 \leq k$ used by the optimal solution (by 'guessing' we mean the exhaustive enumeration of k_0 values, and the output of the maximum obtained solution). Let $\mathcal{M}(C_h)$ be the set of machines that bundle $C_h \in \mathcal{C}^R$ can be assigned to. Formulation (CLP) can be extended as follows:

$$\max \sum_{j \in \mathcal{J}} w_j \left(\sum_{i \in \mathcal{M}} \sum_{k \in S_i} \sum_{C_k :\, j \in C_k} x_{i,C_k} + \sum_{i \in \mathcal{M}} \sum_{C \in \mathcal{C}^R : j \in C} x_{i,C} \right) \text{ s.t.} \tag{CLP+}$$

$$\sum_{k \in S_i} \sum_{C_k} x_{i,C_k} + \sum_{C \in \mathcal{C}^R} x_{i,C} + x_{i,\mathcal{C}_\emptyset} = 1 \qquad \forall i \in \mathcal{M} \tag{11}$$

$$\sum_{i \in \mathcal{M}} \sum_{k \in S_i} \sum_{C_k : j \in C_k} x_{i,C_k} + \sum_{i \in \mathcal{M}} \sum_{C \in \mathcal{C}^R : j \in C} x_{i,C} \leq 1 \qquad \forall j \in \mathcal{J} \tag{12}$$

$$\sum_{i \in \mathcal{M}} x_{i,\mathcal{C}_\emptyset} = m - k_0 \tag{13}$$

$$\sum_{i \in \mathcal{M}(C_h)} x_{i,C_h} = 1 \qquad h = 1, 2, \ldots, l \tag{14}$$

$$\sum_{i=1}^{f} x_{i,\mathcal{C}_\emptyset} = 0 \tag{15}$$

$$x_{i,C} \geq 0 \qquad \forall i, C \tag{16}$$

where (13) implements the k_0-C constraint and (11) has been modified accordingly, (14) implements the RJ constraint for job bundles $C_1, C_2, \ldots, C_l$, and (15) implements the RM constraint for machines $1, 2, \ldots, f$ (wlog we assume that the required machines are the first f of them). Note that in order for (15) to imply that machines $1, \ldots, f$ are used, we are not allowing configurations with no jobs in $\mathcal{C}_i$ for $i = 1, \ldots, f$.

The dual of (CLP+) is

$$\min \sum_{i \in M} y_i + \sum_{j \in \mathcal{J}} z_j + (m - k_0)t + \sum_{h=1}^{l} u_h \text{ s.t.} \tag{DCLP+}$$

$$y_i + \sum_{j \in C_k} z_j \geq w(C_k) \qquad \forall i \in \mathcal{M}, \forall k \in S_i, \forall C_k \in \mathcal{C}(i, k) \tag{17}$$

$$y_i + u_h + \sum_{j \in C_h} z_j \geq w(C_h) \qquad \forall i \in \mathcal{M}, \forall C_h \in \mathcal{C}^R \tag{18}$$

$$t + y_i + v \geq 0 \qquad i = 1, \ldots, f \tag{19}$$

$$t + y_i \geq 0 \qquad i = f+1, \ldots, m \tag{20}$$
$$z \geq 0 \tag{21}$$
$$y, t, v, u \text{ free} \tag{22}$$

The separation oracle of (D-CLP) can be easily extended to solve the new dual LP, and, therefore compute an $(1-\delta)$-approximate solution x^* to (CLP+).

Lemma 2. *There is a polynomial-time $(1-\varepsilon)$-approximate separation oracle for* (DCLP+)*, for any constant $\varepsilon > 0$.*

Proof. There are polynomially many constraints (18)-(20), hence they can be checked for violation in polynomial time. For constraints (17), the proof is similar to Lemma 1, with item values $v_j = w_j - z_j$. The case $y_i \geq 0$ is treated exactly as in Lemma 1.

If $y_i < 0$, then the empty configuration, i.e., picking no items, provides a violating inequality, for $i = f+1, \ldots, n$. For $i = 1, 2, \ldots, f$, where configurations without any jobs are not allowed, there are two cases: (i) If there is an item j with $v_j \geq 0$, then the configuration with this single item provides the violating constraint. (ii) If $v_j < 0,\ \forall j \in \mathcal{J}(i,k)$, then let $v_{max} = \max_j v_j$. If $y_i < v_{max}$ then the configuration with only an item of value v_{max} provides a violated constraint; if $y_i \geq v_{max}$ then there is no configuration that can violate (17), and we move to the next combination of i, k. □

As already mentioned, the dependent rounding of Section 3.2 becomes the rounding procedure of [12] on bipartite graph $G = (U, V, E)$, that satisfies properties (A1), (A2), (A3). We require a rounding of x^* that achieves degree 1 for all vertices in U, degree $m - k_0$ for vertex $C_\emptyset$ in V, and degree 1 for the vertices $\mathcal{C}^R$ in V. The dependent randomized rounding of [12] satisfies these requirements with probability 1 (property (A2)). Theorem 4.4 of [24] expands the negative correlation property (A3) of [12] to all configuration indicator variables of V, stating the following when adapted to our setting:

Theorem 7 ([24, Theorem 4.4])**.** *The indicator random variables X_C of configurations C being picked or not are negatively correlated, and this holds for arbitrary degree bounds on the vertices of V.*

Hence, (9) carries through, and together with the known lower bound for the Maximum Coverage problem [8], we have the following:

Theorem 8. *MMSTC(k-C, RJ, RM) has a polynomial-time $(1 - \frac{1}{e} - \varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$, that satisfies constraints k-C, RJ, RM. This approximation factor is best possible, unless* $\mathsf{P} = \mathsf{NP}$.

Clearly, the bounds of Theorem 8 apply also to MMSTC with any subset of constraints k-C, RJ, RM.

Problem MMSTC(k-C, RM, A). Constraint A allows $l_i \geq 1$ copies of each machine i. In this case, constraint k-C can take two different meanings: (i) It demands that at most k *copies* of machines can be used, or (ii) It demands that at most k *machines* can be used,

i.e., if machine i is not used, then all of its copies are not used, and counts as 1 towards the $m-k$ machines that will not be used. It is easy to see that case (i) can be reduced to problem MMSTC(k-C, RJ, RM) studied above, by simply treating each machine copy as a separate machine. Note that in this case the constraints RJ can also be handled. In this section we assume that constraint k-C has the second meaning, and the handling of constraints RJ will be left as an open problem. The constraint abbreviations are in Table 1.

We will modify (CLP+), so that the configurations for each machine i are combinations of l_i configurations, each in $\mathcal{C}_i$. We also allow for some copies of i to not receive any configuration. Let $\mathcal{P}_i$ be the set of such combined configurations for machine i. Then the Configuration LP becomes

$$\max \sum_{j\in\mathcal{J}} w_j \left(\sum_{i\in\mathcal{M}} \sum_{C\in\mathcal{P}_i : j\in C} x_{i,C} \right) \text{ s.t.} \tag{CLPA}$$

$$\sum_{C\in\mathcal{P}_i} x_{i,C} + x_{i,\mathcal{C}_\emptyset} = 1 \qquad \forall i \in \mathcal{M} \tag{23}$$

$$\sum_{i\in\mathcal{M}} \sum_{C\in\mathcal{P}_i : j\in C} x_{i,C} \le 1 \qquad \forall j \in \mathcal{J} \tag{24}$$

$$\sum_{i\in\mathcal{M}} x_{i,\mathcal{C}_\emptyset} = m - k_0 \tag{25}$$

$$\sum_{i=1}^{f} x_{i,\mathcal{C}_\emptyset} = 0 \tag{26}$$

$$x_{i,C} \ge 0 \qquad \forall i, C \tag{27}$$

and its dual is

$$\min \sum_{i\in M} y_i + \sum_{j\in\mathcal{J}} z_j + (m-k_0)t \text{ s.t.} \tag{DCLPA}$$

$$y_i + \sum_{j\in C} z_j \ge w(C) \qquad \forall i \in \mathcal{M}, \forall C \in \mathcal{P}_i \tag{28}$$

$$t + y_i + v \ge 0 \qquad i = 1, \ldots, f \tag{29}$$

$$t + y_i \ge 0 \qquad i = f+1, \ldots, m \tag{30}$$

$$z \ge 0 \tag{31}$$

$$y, t, v \text{ free} \tag{32}$$

Lemma 3. *There is a polynomial-time* $(1 - \frac{1}{e} - \varepsilon)$*-approximate separation oracle for* (DCLPA)*, for any constant* $\varepsilon > 0$*.*

Proof. The separation oracle can easily check constraints (29), (30). Let us fix a machine $i \in \mathcal{M}$. We explain how the separation oracle can detect whether there is $C \in P_i$ that violates (28).

We define an MMSTC() instance $\mathcal{I}_i$ for machine i as follows. To every job $j \in \mathcal{J}(i)$, we assign a value $v_j := w_j - z_j \leq w_j$, and a size p_{ij}. The set of machines for the instance $\mathcal{I}_i$ is a set of l_i machines, each identical to machine i. The objective is to maximize the total value of the jobs that can be feasibly assigned to these l_i identical machines. Let $J' = \{j \in \mathcal{J}(i) \mid v_j \geq 0\}$. The addition of items with negative values can only decrease the objective function of the MMSTC() subproblem. If $J' = \varnothing$, the maximum-value solution for $\mathcal{I}_i$ consists of a single item, the one with value $v_* = \max_{j \in \mathcal{J}} v_j$, and there is no configuration in $\mathcal{P}_i$ violating (28) iff $y_i \geq v_*$. If $J' \neq \varnothing$ the set of jobs for the instance $\mathcal{I}_i$ is J'.

We distinguish two cases for machine i:

Machine i is not required, i.e., $i \notin \{1, \ldots, f\}$: In this case, instance $\mathcal{I}_i$ is an instance of the MMSTC() problem.

Machine i is required, i.e., $i \in \{1, \ldots, f\}$: In this case, instance $\mathcal{I}_i$ cannot be an instance of the MMSTC() problem, since the solution returned for the MMSTC() subproblem may consist of only empty configurations for *all* l_i copies, and their combination would result in an empty configuration $C \in \mathcal{P}_i$, which is unacceptable. Therefore, we will demand that at least one machine copy (say, the first, since all copies of machine i are identical) has a non-empty configuration, i.e., the first copy treated as a *required machine*. Hence, in this case, instance $\mathcal{I}_i$ is an instance of the MMSTC(RM) problem.

In both cases, an approximate oracle can be implemented to run in polynomial-time by running the MMSTC algorithm of Theorem 8 on instance $\mathcal{I}_i$. It returns l_i configurations, each belonging to $\mathcal{C}_i$. Their combination forms a single configuration $C \in \mathcal{P}_i$ with value v_*.

The oracle concludes that constraints (28) are not violated iff $y_i \geq v_*$. According to Theorem 8, the value v_* returned by the algorithm is at least $(1 - \frac{1}{e} - \varepsilon) \cdot v_{opt}$, where $\varepsilon > 0$ is any constant, and v_{opt} is the optimal value for $\mathcal{I}_i$. Iterating over all machines, we have a $(1 - \frac{1}{e} - \varepsilon)$-approximate separation oracle for (DCLPA). □

Using the approximate separation oracle of Lemma 3 above, Lemma 2.2 in [11] implies that a (fractional) $(1-\frac{1}{e}-\varepsilon)$-approximate solution to (CLPA) can be computed in polynomial time. The application of the dependent rounding of [12] on this approximate fractional solution produces an $((1-\frac{1}{e})^2-\varepsilon)$-approximate integral solution whose properties are codified in the following theorem.

Theorem 9. *MMSTC(k-C, RM, A) has a polynomial-time $((1 - \frac{1}{e})^2 - \varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$, that satisfies constraints k-C, RM, A.*

As was mentioned above, the absence of augmentation constraints A corresponds to the case of $l_i = 1$, $\forall i$. For each machine $i \in \mathcal{M}$, define d_i to be the number of distinct processing times, i.e., $d_i = |\{p_{ij} \mid j \in \mathcal{J}(i)\}|$. In the special case where for every i, l_i and d_i are constant, i.e., are not part of the input, we can avoid using an MMSTC-algorithm as the separation oracle. Instead, for every machine i, we enumerate all possible combinations of l_i submachines from S_i. For each (l_i)-subset T of S_i, assigning a maximum-weight set of jobs to the submachines in T is an instance of the multiple knapsack problem with

assignment restrictions where job j can be assigned only to the machines (knapsacks) in the set $\mathcal{M}(i,j) \cap T$. It is easy to see that for a fixed number of knapsacks and a fixed number of processing times this problem can be solved exactly in polynomial-time (see also Prop. 3 in [6]). Thus we obtain a $(1-\frac{1}{e})$-approximation algorithm for MMSTC(k-C, RM, A):

Corollary 1. *Problem* MMSTC(k-C, RM, A) *where for for all* $i \in \mathcal{M}$, l_i *and* d_i *are fixed has a polynomial-time* $(1-\frac{1}{e})$*-approximation algorithm that satisfies constraints* k-C, RM, A.

4. Minimizing Congestion

In this section we study the SMSTC problem. We provide lower bounds on integrality gaps and a hardness of approximation result. In the instances we will consider, for every job j and for all $i \in \mathcal{M}(j)$, $p_{ij} = 1$. The input is then specified simply as a set $\{u_{ij} \in \mathbb{Z}_{\geq 0} \mid (i,j) \in \mathcal{M} \times \mathcal{J}\}$, and we wish to find the minimum $\rho \geq 1$ for which there is a ρ-feasible assignment for the set $\mathcal{J}$. In Section 4.1 we compare the Configuration LP against a natural linear relaxation. We show integrality gap lower bounds for the Configuration LP and the relaxation of the quadratic program (QP) in Section 4.2. In Section 4.3 we provide a hardness of approximation result.

4.1. The natural formulation versus the Configuration LP

Let us define a natural linear relaxation for MSTC. We use the equivalent SSP problem formulation. We define an LP of the scheduling/facility location variety with assignment variables for pairs of jobs and submachines. Recall that $\bar{u}_{ik}$ denotes the capacity of submachine $k \in S_i$. Variable y_{ik} denotes the extent by which we open submachine $k \in S_i$ and variable x_{ijk} the extent to which job j is assigned to this submachine.

$$\begin{aligned}
&\sum_{i=1}^{m} \sum_{k \in \mathcal{M}(i,j)} x_{ijk} \geq 1 && j \in \mathcal{J} && \text{(LP-natural)}\\
&\sum_{k \in S_i} y_{ik} \leq 1 && i \in \mathcal{M}\\
&\sum_{j \in \mathcal{J}(i,k)} p_{ij} x_{ijk} \leq u_{ik} \cdot y_{ik} && i \in \mathcal{M}, k \in S_i\\
&y_{ik} \geq 0 && i \in \mathcal{M}, k \in S_i\\
&x_{ijk} \geq 0 && i \in \mathcal{M}, j \in \mathcal{J}, k \in \mathcal{M}(i,j)
\end{aligned}$$

It is easy to see that (LP-natural) is a valid relaxation. The YES-instances of MSTC are exactly those for which (LP-natural) has a feasible solution in which all variables take values in $\{0,1\}$.

We denote the Configuration LP for the MSTC search problem as (CCLP). It results from (CLP) by removing the objective function.

$$\sum_{k \in S_i} \sum_{C_k} x_{i,C_k} = 1 \qquad \forall i \in \mathcal{M} \qquad \text{(CCLP)}$$

$$\sum_{i\in\mathcal{M}}\sum_{k\in S_i}\sum_{C_k:j\in C_k} x_{i,C_k} \geq 1 \qquad \forall j \in \mathcal{J}$$
$$x_{i,C_k} \geq 0 \qquad \forall i \in \mathcal{M}, \forall k \in S_i, \forall C_k \in \mathcal{C}(i,k)$$

The upcoming proposition establishes that (CCLP) is stictly stronger than (LP-natural).

Proposition 2. *For every instance $\mathcal{I}$ of MSTC for which* (CCLP) *is feasible,* (LP-natural) *is feasible as well. There is an instance Υ for which* (LP-natural) *is feasible but* (CCLP) *is infeasible.*

Proof. The following claim is straightforward.

Claim 1. *If* (CCLP) *is feasible for instance $\mathcal{I}$,* (LP-natural) *is feasible for $\mathcal{I}$ as well.*

We define the instance Υ. The set $\mathcal{M}$ has cardinality $m \geq 1$. Every machine $i \in [m]$ has two submachines. We define $S_i = \{i_U, i_1\}$ where submachine i_U has capacity $U = m$ and submachine i_1 has capacity 1.

The number n of jobs is $m(U-1)+1$. All jobs have unit processing time on each machine. Each machine $i \in [m]$, has a cluster $\mathcal{J}_i = \{j_l^i \mid l \in [U-1]\}$ of $U-1$ "private" jobs that can only be assigned to i and in particular to both submachines of i. These private clusters account for $m(U-1)$ jobs. The remaining single job out of the n, call it job 1, can be assigned only to submachine i_1, for all $i \in [m]$. Therefore the set of jobs is

$$\mathcal{J} = \{1\} \cup \bigcup_{i=1}^{m} \mathcal{J}_i.$$

Claim 2. (LP-natural) *is feasible for instance Υ.*

Proof of claim. For all i, open submachines i_U and i_1 to the extent of $(U-1)/U$ and $1/U$ respectively. For every $i \in [m]$ and for every job $d = j_l^i \in \mathcal{J}_i$, set $x_{i,d,i_U} = 1$. Therefore each private job of machine i is completely serviced. For every $i \in [m]$, set $x_{i,1,i_1} = 1/U$. Because $U = m$, job 1 is also completely serviced. ∎

Claim 3. (CCLP) *is infeasible for instance Υ.*

Proof of claim. In any solution of (CCLP), any private job from $\mathcal{J}_i$ has to be assigned to machine i to the extent of 1. This can happen either via configurations of size at most U on submachine i_U (type 1 configuration) or configurations of size at most 1 (type 2 configuration) on submachine i_1. The type 2 configurations are partitioned into configurations of type 2a (that do not contain job 1) and configurations of type 2b (that contain job 1). Notice that job 1 can only appear in type 2b configurations. Consider a solution of (CCLP). By feasibility the total number of configurations assigned to all submachines of any machine $i \in [m]$ cannot exceed 1. Let M_1 be the set of machines that are assigned a non-zero fraction of a type 2b configuration. M_1 should be non-empty but then for all $i \in M_1$, we obtain that at least $U-2$ private jobs in in $\mathcal{J}_i$ are not scheduled to the extent of 1. ∎

From Claims 1,2 and 3, the proof is complete. □

4.2. Integrality gaps for SMSTC

Let P be a valid mathematical programming relaxation for computing an assignment for the jobs in $\mathcal{J}$. For $f \geq 1$, we say that P has an *integrality gap of at least f for congestion* if there is an instance $I = \{u_{ij} \mid (i,j) \in \mathcal{M} \times \mathcal{J}\}$ for which P is feasible, but P has no integer feasible solution for any instance $I_\rho = \{\rho \cdot u_{ij} \mid (i,j) \in \mathcal{M} \times \mathcal{J}\}$ with $\rho < f$.

We start by defining an instance Ξ. In this instance each job will have the same capacity on each machine it can be assigned to. The set of machines consists of three blocks of machines, blocks A, B and C. Each machine has two submachines, one with large and one with small capacity. Accordingly we refer to the *big* and the *small* submachine of a given machine. All machines within the same block X have the same large and small capacities at their two submachines. These capacity values are denoted U_X and u_X respectively.

- Block A consists of a single machine with $U_A = 2k$ and $u_A = 2$, where $k \geq 2$ is a positive integer of our choice. We refer to this single machine as machine A.
- Block B consists of 2 machines, B_1 and B_2. The submachine capacities are $U_B = 2k$ and $u_B = 2$.
- Block C consists of 2 machines, C_1 and C_2. The submachine capacities are $U_C = 2k$ and $u_C = 2$.

All jobs have processing time 1. They are partitioned into two sets, those that can only be assigned to small submachines and those that can be assigned to big and small submachines. By slightly abusing terminology we refer to the corresponding sets as *small* and *large* jobs respectively. In what follows when we say that a job may be assigned to the big submachine of machine x it is implied that it can also be assigned to the small submachine of x.

The set of large jobs consists of the disjoint union of two sets F and G. Set F contains $2k$ jobs divided into 2 groups F_1, F_2 each containing k jobs. G consists of k jobs.

- The jobs of F can be assigned to the big submachine of machine A. The jobs of F_i, can be assigned to the big submachine of machine B_i, $i = 1, 2$.
- The jobs of G can be assigned to the big submachines of machines C_1, C_2.

The set of small jobs consists of the disjoint union of two sets P and Q.

- P contains 2 jobs p_1, p_2 that can be scheduled on the small submachine of machine A. Moreover p_i can be assigned on the small submachine of machine C_i, $i \in \{1, 2\}$.
- Set Q contains 1 job, call it q. Job q can be scheduled on the small submachine of machines B_1, B_2.

See Figure 3 for a depiction of the instance Ξ.

Lemma 4. *Linear program* (CCLP) *has a feasible half-integral solution x for the instance Ξ.*

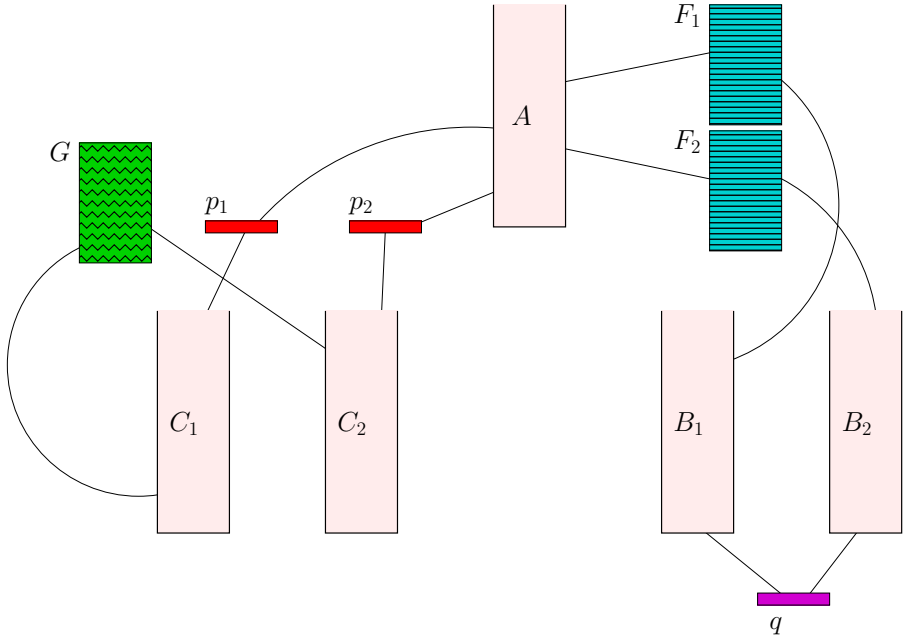


Figure 3: Assignment possibilities for Instance Ξ. The five machines in the instance are A, B_1, B_2, C_1, C_2.

Proof. There is a feasible half-integral solution $\bar{x}$ where to every submachine a configuration of x-value $1/2$ is assigned and each submachine is "open" to the extent of a $1/2$. The jobs in $F \cup G \cup P \cup Q$ can be assigned exactly to two machines each and in $\bar{x}$ they are split equally among these two machines. It is easy to see that these split jobs (of "width" $1/2$ and "height" 1) can be packed into configurations of x-value $1/2$ and height that does not exceed the capacity of the corresponding submachine. □

The reader is reminded that based on the transformation in Section 2 if a job j can be assigned only to a small submachine of machine $X \in \{A, B, C\}$ the tolerance capacity of j on machine X is u_X. If j can be assigned to the large and the small submachine of machine X, the tolerance capacity of j on machine X is U_X.

Lemma 5. *Any integer solution for the instance Ξ that leaves no job unassigned has a congestion of at least $k/4$.*

Proof. There are two possible cases for a feasible solution.

Case 1. There is an $i \in \{1, 2\}$ such that at least half of the jobs in F_i are not scheduled on B_i. I.e., there is an $i \in \{1, 2\}$ such that the big submachine of B_i contains less than half of the jobs in F_i. Therefore at least $k/2$ jobs from F_i are scheduled on machine A. If some job p_j from P is present on A, then p_j experiences a congestion of at least $(k/2)/(u_A) = k/4$. If no job from P is present on A, each machine of block C hosts exactly one job from P. For every $i \in \{1, 2\}$ at least half of the jobs in G must end up on a machine C_{i^*}, for some $i^* \in \{1, 2\}$. The corresponding small job p_{i^*} experiences a congestion of at least $(k/2)/u_C = k/4$.

Case 2. For every $i \in \{1,2\}$, less than half of the jobs in F_i are not scheduled on B_i. I.e., for every $i \in \{1,2\}$ the big submachine of B_i contains at least half of the jobs of F_i. There is an $i^* \in \{1,2\}$ such that the job q is assigned to B_{i^*}. Then this small job experiences a congestion of at least $(k/2)/(u_B) = k/4$. □

The total number n of jobs in the instance Ξ is equal to $3(k+1)$. We have proved the following theorem.

Theorem 10. *The integrality gap of the Configuration LP* (CCLP) *for congestion on an instance with n jobs is at least $(n-3)/12$. This holds even when (i) there are only two distinct capacity values and (ii) every job j has unit processing time, can be assigned to at most two machines and has the same tolerance capacity on every machine in $\mathcal{M}(j)$.*

Define the relaxation (QP-F) of (QP) where the integrality constraints (2) are replaced by

$$x_{ij} \geq 0 \quad \forall i \in \mathcal{M}, \forall j \in \mathcal{J}(i).$$

The instance Ξ we used in Theorem 10 is infeasible for (QP-F). We use the instance Υ defined in the proof of Proposition 2, which we know in turn to be infeasible for (CCLP). In any feasible integer solution, there is a machine $i^* \in [m]$ that processes job 1. In order to service the jobs in J_{i^*} a congestion of U has to be incurred.

There is a feasible fractional solution $\bar{x}$ to (QP-F). For every $i \in [m]$ and for every job $d = j_l^i \in \mathcal{J}_i$, set $\bar{x}_{i,d} = 1$. For every $i \in [m]$, set $\bar{x}_{i1} = 1/U$. Because $U = m$, job 1 is completely serviced. It is easy to see that all capacity constraints are met.

As pointed out by an anonymous reviewer the space of integer solutions of (QP) is unaffected if the quadratic constraints (1) are replaced by the following linear constraints:

$$\sum_{k \in \mathcal{J}(i)} p_{ik} x_{ik} \leq u_{ij} + M(1 - x_{ij}) \quad \forall j \in \mathcal{J}, \forall i \in \mathcal{M}(j) \tag{33}$$

for a big enough M, such as $M \geq \sum_{j \in \mathcal{J}(i)} p_{ij}$. Define (LQP-F) to be the linear relaxation resulting from (QP-F) by replacing the quadratic constraints (1) by (33). The minimum value of M for which this is a valid relaxation is the difference of the two capacity values, i.e., $M = U - 1$. It is easy to see that $\bar{x}$ is feasible for (LQP-F) as well for any $M \geq U - 1$.

Theorem 11. *The integrality gap of (QP-F) and (LQP-F) for congestion is at least m, where m is the number of machines in the instance. This holds even when (i) there are only two distinct capacity values and (ii) every job j has unit processing time and the same tolerance capacity on every machine in $\mathcal{M}(j)$.*

4.3. Hardness of approximation for SMSTC

It is well-known that the following problem is NP-complete [13].

BOUNDED 3D-MATCHING
Input: Set of triples $M \subseteq A \times B \times C$, where A, B, and C are pairwise disjoint sets having the same number q of elements. Every element of $A \cup B \cup C$ occurs in at most 3 triples.
Question: Does M contain a matching, i.e., a subset $M' \subseteq M$ such that $|M'| = q$ and no two elements of M' agree in any coordinate?

Recall Definition 1. An instance of SMSTC which has an f-feasible assignment is called an *f-feasible instance.*

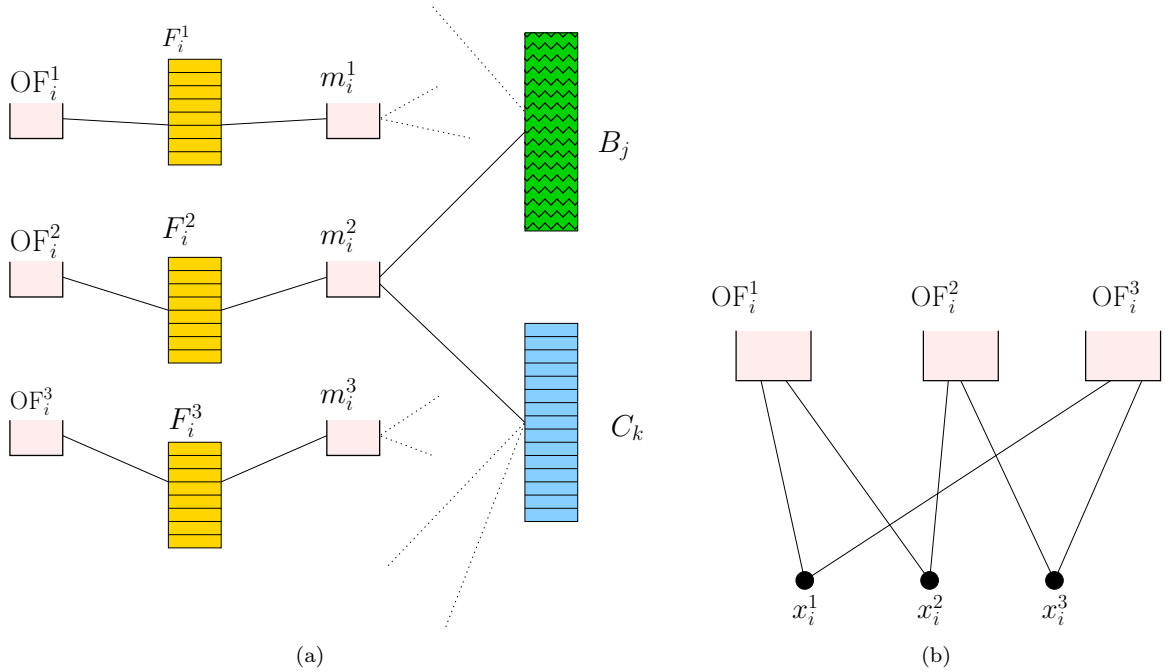


Figure 4: Depiction of the job assignment possibilities for the instance $\phi(I)$. (a) Example of a group $M_i = \{m_i^1, m_i^2, m_i^3\}$ of size 3 and the jobs that can be assigned to the triple-machines in M_i. Machine m_i^2 corresponds to the triple (a_i, b_j, c_k). (b) Allowed assignments for the three placeholder jobs x_i^p, $p \in \{1, 2, 3\}$ to the overfolow machines OF_i^1, OF_i^2, OF_i^3.

Given an instance I of BOUNDED 3D-MATCHING we construct an instance $\phi(I)$ of SMSTC. For every triple in M we have in $\phi(I)$ a dedicated *triple-machine.* For every $i \in [q]$, group together all machines that correspond to triples whose first coordinate is $a_i \in A$ in a group M_i. For every such group, add $|M_i|$ *overflow machines* OF_i^p, $p \in [|M_i|]$. Without loss of generality we may assume that $|M_i| > 1$. The reader is invited to bear in mind from now on that $|M_i| \in \{2, 3\}$. The total number of machines in the instance $\phi(I)$ is $|M| + \sum_{i=1}^{q} |M_i| \leq 12q$.

For $i \in [q]$, we create $|M_i|$ blocks of "dummy" jobs F_i^p, $p \in [|M_i|]$. Every block contains f jobs where $f > 1$ is an integer we will define later. The jobs of block F_i^p can be scheduled

only on machine $m_i^p \in M_i$ and on the overflow machine OF_i^p. The tolerance capacity of every dummy job is f on both machines it can be assigned to. Observe that if a block of dummies ends up on a machine, any additional jobs assigned to the machine will incur congestion larger than 1.

For every $b_j \in B$ we create a set B_j of f' jobs b_j^r, $r \in [f']$, each of which can only be scheduled on the triple-machines that correspond to triples in which b_j is the second component. The quantity f' is an integer larger than f and will be defined later. For every $c_k \in C$ we create a set C_k of f' jobs c_k^r, $r \in [f']$, each of which can only be scheduled on the triple-machines that correspond to triples in which c_k is the third component. The tolerance capacity of these jobs is $2f'$. See Figure 4a.

Finally, there is a set of "placeholder" jobs whose mission will be to block some of the overflow machines. In particular, for every $i \in [q]$, there are d_i placeholder jobs with $d_i = 1$ if $|M_i| = 2$ and $d_i = 3$ if $|M_i| = 3$. The set of placeholder jobs is denoted as $\{x_i^p\}_{p\in[d_i]}$. The assignment possibilities are defined as follows. *Case 1:* $d_i = 1$. The single placeholder job x_i^1 can be assigned to the overflow machines OF_i^p, $p \in \{1, 2\}$, with a tolerance capacity of 1. *Case 2:* $d_i = 3$. Every placeholder job can be assigned to exactly two of the overflow machines OF_i^p, $p \in [3]$, in the way shown in Figure 4b. The tolerance capacity of each placeholder job is 2.

Remark 1. *In a solution to $\phi(I)$ with congestion 1 the following hold for every $i \in [q]$.*
If Case 1 holds for d_i, at least one overflow machine must be reserved exclusively for the placeholder job x_i^1. Therefore at most one dummy block can be assigned to an overflow machine.
If Case 2 holds for d_i, the placeholder jobs must be assigned on at least two overflow machines. No more than a single dummy job may be assigned to an overflow machine that carries a placeholder.

Lemma 6. *If I is a "YES"-instance of* Bounded 3D-Matching*, then $\phi(I)$ is a 1-feasible instance of SMSTC.*

Proof. For all $i \in [q]$ perform the following. Let $\tau = (a_i, b_j, c_k)$ be the triple in the matching that contains element a_i. For all $r \in [f']$ assign the jobs b_j^r and c_k^r to the machine $m_i^p \in M_i$ that corresponds to the triple τ. The dummy jobs of block F_i^p are assigned to the overflow machine OF_i^p. The remaining dummy jobs $F_i^{p'}$, $p' \neq p$, are assigned each to their corresponding machine in M_i. There are $|M_i| - 1$ available overflow machines and we can schedule the d_i placeholders on them. □

Define ρ so that the following relations are satisfied

$$2 \cdot \rho < f \text{ and } \rho \cdot f < f'/3. \tag{34}$$

Lemma 7. *If I is a "NO"-instance of* Bounded 3D-Matching*, then $\phi(I)$ is not a ρ-feasible instance of SMSTC.*

Proof. Assume to the contrary that there is a ρ-feasible assignment σ. Fix a $j \in [q]$. Consider the jobs b_j^r, $r \in [f']$. By the structure of the instance I there are at most 3 triple-machines to which these f' jobs can be assigned. Therefore in σ there is a machine that carries at least $f'/3$ jobs from B_j. By (34) a triple machine that carries even one dummy job can tolerate a total of at most $\rho \cdot f < f'/3$ jobs. Therefore among the triple-machines that can accept the jobs of B_j at least one must have no dummy assigned. The above holds for all j, and in each triple-machine m only jobs from the same set $B_{m(j)}$ can be assigned. Hence there must be at least q machines in M that are empty from dummy so that each receives a job from a distinct B_j, $j \in [q]$.

Claim 4. *In a ρ-feasible assignment σ, for every $i \in [q]$, exactly one triple-machine in group M_i is empty from dummy.*

Proof of Claim. We have shown that at least q machines in M must be empty from dummy. We will show that for each i at most one machine in M_i can be empty from dummy. This will establish the claim. We distinguish two cases.

Case 1: $d_i = 1$. Assume that in σ both machines m_i^1, m_i^2 in M_i are empty from dummy. Then the two overflow machines $\mathrm{OF}_i^1, \mathrm{OF}_i^2$ take each one block of dummy jobs. The placeholder job x_i^i must live on the same machine with f other jobs. By (34) this incurs a congestion larger than ρ, a contradiction.

Case 2: $d_i = 3$. Assume that in σ at least two among the three machines in M_i are empty from dummy. Then at least two overflow machines $\mathrm{OF}_i^p, \mathrm{OF}_i^{p'}$ take each one block of dummy jobs. At least one of the three placeholder jobs x_i^1, x_i^2, x_i^3 must live in σ on the same machine with f other jobs. By (34) this incurs a congestion larger than ρ, a contradiction. The proof of the claim is complete.

By Claim 4, exactly one machine from each of the q groups M_i must be empty from dummy. Let M' be the set of these machines. Each machine in M' gets at least one member from a distinct B_j. M' induces a 2D perfect matching of $A \times B$.

Similarly, a triple machine that carries a dummy job can tolerate at most $\rho \cdot f < f'/3$ jobs from C_k, for any $k \in [q]$. Since σ is ρ-feasible, for every $k \in [q]$ at least one job from C_k is assigned to a machine that is empty of dummy, i.e., to a machine of M'. Clearly, no two jobs from different C_k, $C_{k'}$ sets, with $k \neq k'$, can appear on the same triple-machine. The q machines of M' that are empty from dummy induce a feasible 3D-Matching of $A \times B \times C$. We have reached a contradiction.

As long as f' is larger than a suitable constant, setting $\rho = \frac{\sqrt{f'}}{4}$ and $f = \lceil \sqrt{f'}/2 \rceil + 1$ satisfies (34). □

We conclude that unless $\mathsf{P} = \mathsf{NP}$ there is no polynomial-time algorithm that on input $\phi(I)$ can output a solution with congestion at most $\sqrt{f'}/4$ times the optimum. Given that the number n of jobs in $\phi(I)$ is equal to $2f'q + f \cdot |M| + \sum_{i=1}^{q} d_i$ and that $q \leq |M| \leq 9q$, we have that $n = \Theta(f'q)$. To keep the reduction polynomial-time it must be that $f' = O(q^c)$ for some constant $c > 0$. In other words, $f' = n^{1-\varepsilon}$ for an arbitrary constant $\varepsilon > 0$ of our choice. The following theorem has been proved.

Theorem 12. *For any constant $\varepsilon > 0$, there is no polynomial-time $(n^{1/2-\varepsilon})$-approximation algorithm for SMSTC, unless* $\mathsf{P} = \mathsf{NP}$. *This holds even when (i) there are only four distinct capacity values and (ii) every job j has unit processing time, can be assigned to at most three machines and has the same tolerance capacity on each machine in $\mathcal{M}(j)$.*

5. Conclusions

Several open problems arise from our work. Closing the $n^{1/2}$-gap between Theorems 12 and 10 or coming up with a $n^{1-\varepsilon}$-approximation algorithm for some constant $0 < \varepsilon < \frac{1}{2}$ is the obvious open question for SMSTC. Another open problem is the improvement of the $(1-\frac{1}{e})^2-\varepsilon$ factor for MMSTC(k-C, RM, A) (Theorem 9). An approximation factor of $1-\frac{1}{e}-\varepsilon$ may be possible, and it can be achieved if there is an $(1+\varepsilon)$-approximate separation oracle for (DCLPA), e.g., if there is a PTAS for MMSTC() (or, even better, MMSTC(RM)) for the special case where all machines are *identical*. Improving the factor of Theorem 6 or Theorem 9 in the case of identical machines is another open problem. In fact, achieving better approximation factors for other restricted versions of our problems (e.g., having only $O(\log n)$ different tolerance capacity values, or when the number of machines is constant) is an interesting research direction. An intermediate goal can be the extension of the $(1-\frac{1}{e})^2-\varepsilon$ approximation factor to include constraints RJ, i.e., to MMSTC(k-C, RJ, RM, A), which would cover the combination of all our constraints.

Dealing with additional constraints to those that were considered here is another possible direction. An immediate extension is the introduction of a budget constraint. Given assignment costs c_{ij} for all jobs j on machines i, and a budget B, the total cost of scheduled jobs cannot exceed the budget B. This can be handled by the extension of the Chernoff bounds in [22] as used in [18] and [5]. The work of [2, 5] may be useful in order to include additional matroidal constraints to our problems, or to deal with more general objectives, e.g., when the weights w of the jobs depend also on the machine they are scheduled on.

Acknowledgements

We thank the anonymous reviewers, whose comments greatly improved the completeness and the presentation of this work. The first author also thanks Terry Todd and Dongmei Zhao for introducing him to the time-sharing problem arising within the Digital Twins context.

References

[1] N. Bansal, M. Sviridenko, The Santa Claus problem, in: Proceedings of the 38th annual ACM symposium on Theory of Computing (STOC), ISBN 1-59593-134-1, 31–40, 2006.

[2] G. Călinescu, C. Chekuri, M. Pál, J. Vondrák, Maximizing a Monotone Submodular Function Subject to a Matroid Constraint, SIAM J. Comput. 40 (6) (2011) 1740–1766.

[3] C. Chekuri, S. Khanna, A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem, SIAM J. Comput. 35 (3) (2005) 713–728.

[4] C. Chekuri, A. Kumar, Maximum Coverage Problem with Group Budget Constraints and Applications, in: APPROX-RANDOM, vol. 3122 of *LNCS*, Springer, 72–83, 2004.

[5] C. Chekuri, J. Vondrák, Randomized Pipage Rounding for Matroid Polytopes and Applications, CoRR abs/0909.4348.
[6] G. Dahl, N. Foldnes, LP based heuristics for the multiple knapsack problem with assignment restrictions, Ann. Oper. Res. 146 (1) (2006) 91–104.
[7] K. Eisemann, The trim problem, Management Science 3 (3) (1957) 279 – 284.
[8] U. Feige, A Threshold of ln n for Approximating Set Cover, J. ACM 45 (4) (1998) 634–652.
[9] U. Feige, J. Vondrák, Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e, in: 47th IEEE Symposium on Foundations of Computer Science (FOCS, 667–676, 2006.
[10] A. M. C. Ficker, F. C. R. Spieksma, G. J. Woeginger, The transportation problem with conflicts, Ann. Oper. Res. 298 (1) (2021) 207–227.
[11] L. Fleischer, M. X. Goemans, V. S. Mirrokni, M. Sviridenko, Tight Approximation Algorithms for Maximum Separable Assignment Problems, Math. Oper. Res. 36 (3) (2011) 416–431.
[12] R. Gandhi, S. Khuller, S. Parthasarathy, A. Srinivasan, Dependent rounding and its applications to approximation algorithms, J. ACM 53 (3) (2006) 324–360.
[13] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, 1979.
[14] P. C. Gilmore, R. E. Gomory, A linear programming approach to the cutting-stock problem, Operations Research 9 (1961) 849 – 859.
[15] K. Jansen, L. Rohwedder, A Quasi-Polynomial Approximation for the Restricted Assignment Problem, SIAM J. Comput. 49 (6) (2020) 1083–1108.
[16] N. Karmarkar, R. Karp, An efficient approximation scheme for the one-dimensional bin-packing problem, in: Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), 312–320, 1982.
[17] K.-T. Ko, B. Davis, Delay Analysis for a TDMA Channel with Contiguous Output and Poisson Message Arrival, IEEE Trans. on Communications 32 (6) (1984) 707–709.
[18] A. Kulik, H. Shachnai, T. Tamir, Maximizing submodular set functions subject to multiple linear constraints, in: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009, SIAM, 545–554, 2009.
[19] J. K. Lenstra, D. B. Shmoys, E. Tardos, Approximation Algorithms for scheduling unrelated parallel machines, Mathematical Programming A 46 (1990) 259–271.
[20] R. Minerva, G. M. Lee, N. Crespi, Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models, Proceedings of the IEEE 108 (10) (2020) 1785–1824.
[21] S. M. A. Oteafy, Resource augmentation in Heterogeneous Internet of Things via UAVs, in: 2021 IEEE Global Communications Conference (GLOBECOM), 1–6, 2021.
[22] A. Panconesi, A. Srinivasan, Randomized Distributed Edge Coloring via an Extension of the Chernoff-Hoeffding Bounds, SIAM J. Comput. 26 (2) (1997) 350–368.
[23] T. Rothvoss, Better Bin Packing Approximations via Discrepancy Theory, SIAM J. Comput. 45 (3) (2016) 930–946.
[24] B. Saha, A. Srinivasan, A new approximation technique for resource-allocation problems, Random Struct. Algorithms 52 (4) (2018) 680–715.
[25] A. Sharma, E. E. Kosasih, J. Zhang, A. Brintrup, A. Calinescu, Digital Twins: State of the Art Theory and Practice, Challenges, and Open Research Questions, CoRR abs/2011.02833.
[26] A. Silberschatz, P. B. Galvin, G. Gagne, Operating System Concepts, Wiley Publishing, 9th edn., 2012.
[27] A. Srinivasan, Distributions on Level-Sets with Applications to Approximation Algorithms, in: Proceedsings of 42nd IEEE Annual Symposium on Foundations of Computer Science (FOCS), 588–597, 2001.
[28] O. Svensson, Santa Claus Schedules Jobs on Unrelated Machines, SIAM J. Comput. 41 (5) (2012) 1318–1341.
[29] S. Verma, Y. Kawamoto, Z. Fadlullah, H. Nishiyama, N. Kato, A Survey on Network Methodologies for Real-Time Analytics of Massive IoT Data and Open Research Issues, IEEE Communications Surveys Tutorials 19 (3) (2017) 1457–1477.

[30] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, S. Ulukus, Age of Information: An Introduction and Survey, IEEE Journal on Selected Areas in Communications 39 (5) (2021) 1183–1210.