Scheduling Non-Unit Jobs to Minimize Calibrations

Jeremy T. Fineman
Georgetown University
jfineman@cs.georgetown.edu

Brendan Sheridan
Georgetown University
bss45@georgetown.edu

ABSTRACT

The recently proposed Integrated Stockpile Evaluation (ISE) problem extends a classic offline scheduling problem where n jobs, each with arrival times and deadlines, must be scheduled nonpreemptively on m machines. The additional constraint in the ISE problem is that a machine may only be used if it has been calibrated recently. The goal is to minimize the number of calibrations necessary to complete all the jobs before their deadlines.

This paper presents a good polynomial-time approximation algorithm for the ISE problem general case where each job may have a different processing time. (Prior work was restricted to unit processing times.) The ISE problem generalizes a classic interval-scheduling problem where the goal is to minimize the number of machines. We show constructively that the other direction is also true, i.e., that the interval-scheduling bounds are also achievable. Specifically, suppose we have a black-box interval scheduling algorithm that finds an s-speed αm-machine solution to the interval scheduling problem. Then our ISE algorithm finds an $O(\alpha)$-approximation for number of calibrations using $O(\alpha m)$ machines with s-speed augmentation.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Sequencing and scheduling*

Keywords

Integrated Stockpile Evaluation; approximation algorithms; calibration; resource allocation; scheduling

1. INTRODUCTION

The ***Integrated Stockpile Evaluation (ISE)*** problem is an offline multi-machine scheduling problem, recently introduced by Bender et al. [5]. What distinguishes the ISE problem is that a machine is unusable unless it undergoes a calibration, and jobs may only be scheduled on a machine if that machine has been calibrated recently. Specifically, if a calibration is performed on a machine at time t, then the machine remains usable or ***calibrated*** for the interval $[t, t+T)$, for fixed T. Each machine may be calibrated multiple times, but the machine must remain idle between its calibrated intervals. Calibrations are instantaneous, so it is feasible to calibrate the machine at times 0, T, $2T$, and so on. Although calibrations have no temporal cost, they are considered the expensive feature of a solution.

More precisely, the ISE problem (denoted $P|r_j, d_j|$#calibrations in standard scheduling notation [10]) is defined as follows [5]. The input consists of a set J of n jobs, an integer number m of identical machines, and an integer $T \geq 2$ specifying a ***calibration length***. Each job j has a processing time $p_j \leq T$, a release time r_j, and a deadline $d_j \geq r_j + p_j$. A schedule is feasible if it schedules jobs nonpreemptively[1] on machines such that 1) every job completes before its deadline, and 2) every job is scheduled without preemption completely within a single calibrated interval. The goal is to find a feasible schedule that minimizes the number of calibrations performed.

The ISE problem formalizes scheduling issues that arise as part of a nuclear-weapons-testing program with the same name at Sandia National Laboratories [6]. The high-level goal is to perform tests on a set of nuclear weapons to verify their integrity, with the constraint that the testing devices be frequently re-calibrated to guarantee accurate results. See [5, 6] for more details.

Bender et al. [5] give the first algorithms for the ISE problem. They study the restricted case that for all jobs j, $p_j = 1$. Even in this special case of unit processing times, the problem is nontrivial. Bender et al. [5] give two greedy scheduling algorithms. Their first algorithm guarantees an optimal schedule (i.e., minimizing the number of calibrations) whenever a 1-machine schedule is feasible. Their second algorithm gives a 2-approximation for the multi-machine case.

This paper addresses the ISE problem for non-unit processing times, which Bender et al. [5] leave as an open problem. It is not hard to see that testing whether a feasible schedule exists is NP-hard by a reduction from Partition. (Use $m = 2$ machines, and assign all jobs the same release time $r_j = 0$ and deadline $d_j = T$.) Thus ignoring the goal of minimizing calibrations, obtaining a polynomial-time algorithm that finds any feasible schedule necessitates *resource augmentation* [12], where the algorithm is given more or faster machines than the optimal solution against which it is compared. While the notion of resource augmentation was introduced as a technique for producing online algorithms, it has also become commonplace in recent offline scheduling literature.

More precisely, we define resource augmentation as follows. Let $\mathrm{OPT}(I)$ denote the best possible value of the objective function

[1]Nonpreemptive means that when job j is scheduled on a machine, it must be scheduled for p_j consecutive timesteps.

SPAA'15, June 13–15, 2015, Portland, OR, USA.

ACM 978-1-4503-3588-1/15/06 ...$15.00.
DOI: http://dx.doi.org/10.1145/2755573.2755605.

(i.e., number of calibrations) over feasible m-machine schedules of instance I. Adopting terminology introduced by Phillips et al. [13], we say that an algorithm is a ***w-machine s-speed ρ-approximation algorithm*** if it always achieves a feasible schedule with value at most $\rho\text{OPT}(I)$ given wm machines each operating s times faster.

To understand how good a solution we can expect for the ISE problem, observe that the ISE problem extends the classic machine-minimization (MM) problem: given a set of jobs with release times, deadlines, and processing times, find the minimum number of machines necessary to schedule all jobs by their deadlines. Specifically, given an instance to MM, construct an ISE instance by setting $T = \max_j \{d_j\} - \min_j \{r_j\}$. A w-machine s-speed solution to the ISE problem (with any approximation quality) would yield an s-speed w-approximation to the MM problem. Thus if the best s-speed approximation algorithm to the MM problem uses α times the optimum number of machines, then the ISE problem requires α-machine augmentation when limited to s-speed augmentation. Similarly, because the number of machines and the number of calibrations are identical in this construction, the best s-speed α-machine approximation we can expect for the ISE problem has an approximation ratio $\rho \geq \alpha$.

Contributions

Our main result is an algorithm that uses any MM algorithm as a black box, with only a constant factor overhead in terms of machine augmentation. Specifically, suppose we have an s-speed α-approximation algorithm for the MM problem. Then our algorithm results in an $O(\alpha)$-machine s-speed $O(\alpha)$-approximation to the ISE problem. As explained above, this is the best we can expect to within constant factors.

To understand this result concretely, let us consider the current best approximation algorithms for the MM problem. Chuzhoy et al. [8] give an $O(\text{OPT})$-approximation for the MM problem, meaning that the solution uses $O(\text{OPT}^2)$ machines. If $\text{OPT} = O(1)$, then their algorithm is a 1-speed $O(1)$-approximation algorithm. More generally, combining their solution with Raghavan and Thompson's previous best $O(\log n/\log\log n)$-approximation [14] yields an $O(\sqrt{\log n/\log\log n})$-approximation for the MM problem for arbitrary OPT. Both of these results use no speed augmentation and apply to the general case.

In more recent work, Bansal et al.[2] study the case that $\text{OPT} = 1$, giving an $O(1)$-speed 1-approximation (i.e., a one-machine solution) for this special case. Unfortunately, it is not clear how to generalize this result past 1 machine. Im et al. [11] give an algorithm guaranteeing either a $(1+\varepsilon)$-speed 2-approximation or a $(2+\varepsilon)$-speed 1-approximation for the MM problem for any OPT. But unlike the preceding results, their algorithm runs in *quasi-polynomial time*, i.e., $O(n^{O(\log^c n)})$ for some $c > 0$, not polynomial time.

Combining our algorithm with the above MM algorithms, we get the following concrete results:

- A polynomial-time $O(\sqrt{\log n/\log\log n})$-machine 1-speed algorithm giving an $O(\sqrt{\log n/\log\log n})$ approximation for the ISE problem. That is, given an ISE instance that is feasible using m machines and C calibrations, our algorithm produces a solution using $O(C\sqrt{\log n/\log\log n})$ calibrations on $O(m\sqrt{\log n/\log\log n})$ machines.
- Whenever the input instance is feasible on $O(1)$ machines, then we have a polynomial-time $O(1)$-machine 1-speed $O(1)$-approximation for the ISE problem. More generally, if the instance is feasible on m machines, then we get a 1-speed $O(m)$-machine $O(m)$-approximation, which is better than the first bound if $m = o(\sqrt{\log n/\log\log n})$.
- A quasi-polynomial-time $O(1)$-machine $O(1)$-speed $O(1)$-approximation for the ISE problem.

2. ALGORITHM OVERVIEW

This section gives our top-level algorithm for the ISE problem. The high-level algorithm is simple: partition the input jobs J into two subsets J_{long} and J_{short}, each containing the jobs with "long" and "short" windows respectively. Schedule those jobs independently, on disjoint machines, using the specialized algorithms described in Sections 3 and 4. The partitioning itself is trivial, and this process at most doubles the number of calibrations and machines beyond either of the algorithms.

More precisely, we define long and short jobs as follows. Note that the definition of long and short is based on the job's window (release time and deadline), not its processing time.

DEFINITION 1. *We say that job j is **long** or a **long-window job** if $d_j - r_j \geq 2T$. We say that a job j is **short** or a **short-window job** if $d_j - r_j < 2T$.*

Our algorithm for long-window jobs uses an integer-program relaxation followed by a greedy rounding procedure (Section 3). Assuming all jobs have long windows, our algorithm yields an $O(1)$-machine 1-speed $O(1)$-approximation to the ISE problem. The algorithm can also be modified to trade more speed for fewer machines, giving a 1-machine $O(1)$-speed $O(1)$-approximation. These bounds are better than those stated in the introduction with regard to the approximation ratio on number of calibrations and the amount of machine-augmentation employed. The higher overheads arise in scheduling short-window jobs (Section 4), where we apply an MM algorithm as a black box. The short-window algorithm also increases the number of machines and the approximation ratio by a constant factor. Unfortunately, for small-window jobs (as for the MM problem in general), it is not clear how to trade speed for machines.

Combining all of these results, we get our main theorem, whose proof follows directly from Theorems 12 and 20.

THEOREM 1. *Suppose there is an s-speed $O(\alpha)$-approximation algorithm $\mathcal{A}$ for the MM problem. Then using $\mathcal{A}$ as a black box, our algorithm is an $O(\alpha)$-machine s-speed $O(\alpha)$-approximation for the ISE problem. Moreover, the running time of our algorithm is a polynomial in the length of the input (i.e., polynomial in n and the precision of other numbers), multiplied by the running time of $\mathcal{A}$.*

3. SCHEDULING LONG-WINDOW JOBS

This section focuses on the special case of ISE where all jobs have long windows. For this special case, our main algorithm yields a 1-speed $O(1)$-machine $O(1)$-approximation, and we also show that this solution can be transformed into an $O(1)$-speed 1-machine $O(1)$-approximation. Intuitively, long jobs are easier to cope with than short jobs because they have more options on where to be scheduled, but capturing this intuition is not trivial.

We begin this section with one key insight: introducing an extra restriction to the long-window ISE problem makes it easier to solve, without significantly compromising the quality of solution. Specifically, we focus on what we call the ***trimmed ISE (TISE)*** problem. The TISE problem is exactly the same as the ISE problem, except that there is one additional restriction on the schedule: a job may be scheduled only inside a calibration that falls completely within the job's window. Said differently, consider a calibration starting at time t, spanning the time interval $[t, t+T)$. Job j, with window $[r_j, d_j)$, may only be scheduled in this calibration if

$r_j \le t \le d_j - T$. We call this extra restriction the ***TISE restriction*** or ***TISE constraint***. Note that the TISE constraint is specific to long jobs because jobs with windows shorter than T are infeasible in the TISE problem.

The main advantage of the TISE problem is that whenever a job is scheduled within a particular calibration, it would be feasible to schedule the job anywhere within that calibration, which gives us flexibility in the schedule. Moreover, because of this flexibility, given an assignment of jobs to calibrations we can infer a schedule. We thus need only focus on 1) finding a schedule of calibrations, and 2) finding an assignment of jobs to those calibrations.

The bulk of this section gives an algorithm for the TISE problem, which includes several steps. First, we construct a linear-programming (LP) relaxation of the TISE problem. The LP allows for both fractional calibrations and fractional job assignments. We then perform a greedy rounding step that yields an integer calibration schedule such that a fractional assignment of jobs to calibrations remains feasible. This rounding step increases the number of calibrations and machines by a constant factor. Note that fractional job assignments correspond to a preemptive schedule, whereas integer job assignments correspond to a nonpreemptive schedule. Finally, we convert the preemptive schedule to a nonpreemptive schedule, using a constant factor extra machines and calibrations, through a variant of earliest deadline first (EDF) scheduling. Earliest deadline first does not generally work for nonpreemptive scheduling with arbitrary release times, deadlines, and processing times, but the TISE restriction helps us here.

A TISE solution is good enough

Because the TISE problem is more restricted, any valid TISE schedule is also a valid ISE schedule. The question is what happens to the quality of the solution. We argue in the following lemma that the TISE solution is as good, to within constant factors. Specifically, the optimal TISE solution uses at most three times the number of machines and calibrations as the optimal ISE solution. It thus suffices to solve a TISE problem on $m' = 3m$ machines.

The proof of the following lemma leverages the definition of long-window jobs. In particular, in order for the presented construction to apply, the threshold for being long must be at least $2T$. This proof is the reason for the choice of constant in Definition 1. (Making the threshold larger is okay, but that would weaken the bounds for short-window jobs.)

LEMMA 2. *Consider any long-window ISE instance. Suppose that there exists a feasible ISE schedule using m machines and C calibrations. Then there exists a feasible TISE schedule using at most $m' = 3m$ machines and $3C$ calibrations.*

PROOF. The proof is by construction, which is illustrated by Figure 1. Consider any machine i in the ISE schedule. We shall use three machines, denoted i', i^+, and i^-, in the TISE schedule. For a calibration starting at time t on machine i in the ISE schedule, we create three calibrations in the TISE schedule: a calibration on machine i' at time t, a calibration on machine i^+ at time $t+T$, and a calibration on machine i^- at time $t-T$. Because the calibrations on machine i^+ and i^- are T-step translations of the calibrations on i, the calibrations themselves are feasible.

We next transform the schedule of jobs. Consider each job j in the ISE schedule. Let x_j be the job's start time in the ISE schedule. Let i be the machine on which the job is scheduled. Let t_j be the start time of the calibration containing the job, i.e., such that $x_j \in [t_j, t_j + T)$. If $r_j \le t_j \le d_j - T$, i.e., the job is already feasibly scheduled with regards to the TISE restriction, then schedule the job at time x_j on machine i'. If $r_j > t_j$, then delay the job, scheduling it at time $x_j + T$ on machine i^+. If $d_j < t_j + T$, then advance the job, scheduling it at time $x_j - T$ on machine i^-.

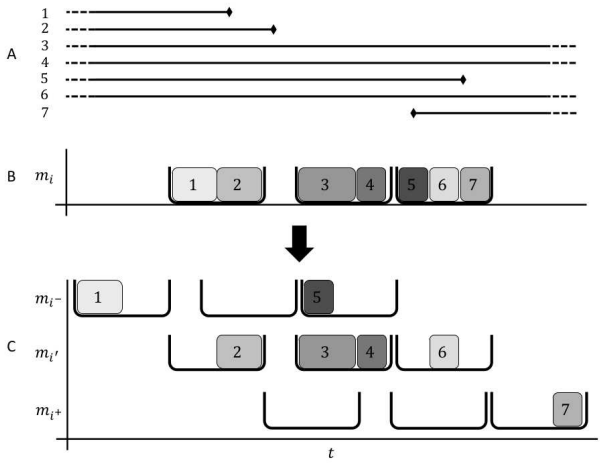

Figure 1: Example of the transformation from a feasible ISE schedule on one machine to a feasible TISE schedule on 3 machines with $3\times$ the calibrations, as described in proof of Lemma 2. (A) indicates the job windows for the relevant long-window jobs, with the endpoints of line j corresponding to r_j and d_j. (B) shows the original feasible schedule on machine i, and (C) shows the constructed TISE schedule given by the proof, where the buckets are calibrations and the shaded rectangles are jobs—the width of rectangle j is its processing time p_j. Jobs 1 and 5 are moved to the advanced calibrations on i^- because their deadlines fall within the original calibration. Similarly, job 7 is moved to the delayed calibrations on i^+ because its release time falls within the original calibration.

We must show that delaying or advancing a job enforces the TISE restriction. Consider a delayed job j, i.e., one with $r_j > t_j$ in the ISE schedule. Because the job is long, we have $d_j \ge r_j + 2T > t_j + 2T$. Thus the calibration $[t_j + T, t_j + 2T)$ is contained fully within j's window. Moreover, the job must complete by time $t_j + T$ in the ISE schedule, so it must also complete by time $t_j + 2T$ in the TISE schedule. A similar argument applies to advanced jobs.

Finally, because the ISE schedule is a nonpreemptive schedule such that no two jobs run at the same time on the same machine, and each of the machines i', i^+, and i^- only receives a subset of jobs from the ISE schedule on i (all translated by 0, $+T$, or $-T$ timesteps), the TISE schedule is also a valid nonpreemptive schedule. □

Polynomially many calibration points suffice

When constructing scheduling LPs, it is common practice to use variables indexed by each timestep and then argue after the fact that the LP can be transformed to one where the number of variables is polynomial in n. This approach, however, may require that time be discrete, whereas our TISE problem statement does not require that release times, processing times, or deadlines be integers. We thus determine what the important times are up front before constructing the LP. The following lemma states that there are only n^2 times that matter.

LEMMA 3. *There exists an optimal solution to the TISE problem such that the following holds for every calibration on every machine i. If a calibration is made at time t on machine i, then*

Algorithm 1: Rounding calibrations produced by the LP

```
carryover = 0 ;        // carried calibration fraction
C = ∅ ;                // new calibration schedule
i = 0 ;                // machine number
foreach t ∈ T in increasing order do
    carryover = carryover + C_t
    while carryover ≥ 1/2 do
        add a calibration at time t on machine i to C
        carryover = carryover − 1/2
        i = i + 1  mod 3m'
```

either t is equal to the release time of a job, or the calibration immediately follows the preceding calibration on that machine (i.e., there is a calibration at time $t-T$ on machine i).

PROOF. Consider an optimal schedule for the TISE problem. We can iteratively transform it to one that obeys the lemma as follows. Consider the calibrations on each machine in increasing order of time. If the kth calibration does not obey the lemma, then decrease its start time (and the corresponding start time of any jobs therein) until the calibration's start time hits a release time or the end of the $(k-1)$th calibration, whichever comes first. Because the calibration is not moved past any release times, all jobs in the calibration can be advanced without sacrificing feasibility. □

Because an optimal solution does not use any empty calibrations, the lemma implies that there are at most n^2 possible calibrations on each machine. Specifically, there may be a calibration at any release time. There may also be calibrations packed immediately after this one, but there can only be n such calibrations.

We define $\mathcal{T} = \{r_j + kT | j \in J, k \in \{0,1,2,\ldots,n\}\}$ as the set of ***potential calibration points***.

A linear-program for the TISE problem

The goal of our linear program (LP) is to determine a schedule of calibrations on machines and an assignment of jobs to those calibration points. (Recall that for the TISE problem, a full schedule can be inferred by an assignment of jobs to calibrations because the jobs can, by definition, be scheduled in any order.) An integer solution to the LP corresponds to a feasible TISE schedule. We will start from a fractional solution and round it.

Before stating the LP, let us mention two simplifying ideas. It should be clear that both of the simplifications can only improve the value of the optimal solution because feasible TISE schedules can be trivially transformed into LP solutions.[2] First, our LP ignores the mapping of calibrations to machines, instead only requiring that at most m' calibrations overlap at any time. Second, our LP groups calibrations by time, ignoring how jobs are partitioned across same-time calibrations.

Leveraging these simplifications, our LP has two types of variables. The variable C_t denotes the number of calibrations made at time t. The variable X_{jt} indicates whether (or how much of) job j is assigned to the calibrations at time t. In both cases, following from Lemma 3, we use the restriction that $t \in \mathcal{T}$ be one of the potential calibration points.

[2]In fact, one can argue that for the fractional solution, the value of the optimal solution is unchanged. For an integer solution, however, it may not be feasible to produce a TISE schedule from the integer solution.

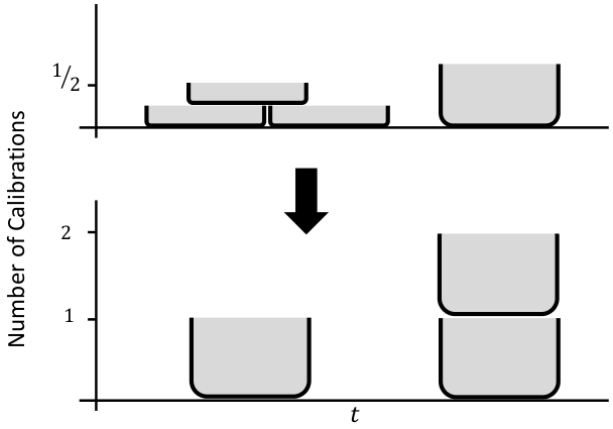

Figure 2: Example calibration rounding following Algorithm 1. Buckets indicate scheduled calibrations and their height represents the amount of calibration. Calibration points are reached after the second and fourth fractional calibrations, resulting in a full calibration and two full calibrations respectively.

We are left with the following LP relaxation of the TISE problem:

$$\text{minimize} \quad \sum_{t\in\mathcal{T}} C_t$$

$$\text{subject to} \quad \sum_{t'\in\mathcal{T},\, t-T<t'\le t} C_{t'} \le m' \qquad \forall t \in \mathcal{T} \quad (1)$$

$$X_{jt} \le C_t \qquad \forall j \in J_{long}, t \in \mathcal{T} \quad (2)$$

$$\sum_j X_{jt} p_j \le C_t T \qquad \forall t \in \mathcal{T} \quad (3)$$

$$\sum_{t\in\mathcal{T}} X_{jt} = 1 \qquad \forall j \in J_{long} \quad (4)$$

$$X_{jt} = 0 \qquad \forall j \in J, t \in \mathcal{T} \text{ s.t. } t < r_j \text{ or } t+T > d_j \quad (5)$$

$$X_{jt}, C_t \ge 0 \qquad \forall j \in J_{long}, t \in \mathcal{T} \quad (6)$$

The first constraint guarantees that there are not more than m' calibrations at any timestep. The second constraint says that each job can only be assigned to each calibration once (or more accurately, the fraction of a job assigned to a calibration point cannot exceed the fraction of calibrations performed at that point). The third constraint enforces that the total work assigned to a calibration point (i.e., the fraction of jobs times their processing time) be at most the total processing power of the calibration point (i.e., the number of calibrations times T). The fourth constraint requires that every job be scheduled completely. The fifth constraint enforces the TISE restriction, that jobs only be assigned to calibrations that are contained in their windows. Finally, the last constraint is a nonnegativity constraint on job assignments and calibrations.

The TISE algorithm

As noted previously, our TISE algorithm has three steps. First, we solve the LP relaxation for $m' = 3m$ machines. The LP solution could have both fractional calibrations C_t and fractional job assignments X_{jt}. Second, we apply a simple greedy-rounding algorithm, given by Algorithm 1, to produce an integer calibration schedule on $3m'$ machines. The rounding algorithm scans calibrations C_t in order of time, keeping a running total. Whenever the total reaches the next multiple of $1/2$, the algorithm creates 1 new calibration at

Algorithm 2: Assign jobs J_{long} given calibration schedule $\mathcal{C}$

mirror the calibration schedule $\mathcal{C}$ on twice as many machines
let $\mathcal{C}'$ be the resulting calibration schedule

foreach *calibration in* $\mathcal{C}'$ *in nondecreasing order of time* **do**
  let t be the start time of the calibration
  $used = 0$; `// work in calibration`
  let $J' = \{j \in J_{long} | j$ unscheduled and $r_j \le t \le d_j - T\}$
  let $j \in J'$ be a job with earliest deadline
  while $j \neq$ NULL *and* $p_j + used \le T$ **do**
    schedule job j at time $t + used$ in the calibration
    $used = used + p_j$
    remove j from J'
    let $j \in J'$ be a job with earliest deadline

Algorithm 3: Augmented calibration-rounding procedure used only for the proof of Lemma 5 and Corollary 6

$carryover = 0$; `// carried calibration fraction`
set $y_j = 0$ for all j ; `// carried job fractions`
$\mathcal{C} = \emptyset$; `// new calibration schedule`
foreach $t \in \mathcal{T}$ *in increasing order* **do**
  while $carryover + C_t \ge 1/2$ **do**
    create a calibration at time t in $\mathcal{C}$
    $frac = \frac{1/2 - carryover}{C_t}$; `// take part of` C_t
    $carryover = carryover + frac \cdot C_t$
    foreach *job* j **do**
      $y_j = y_j + frac \cdot X_{jt}$
      $X_{jt} = X_{jt} - frac \cdot X_{jt}$
      if $r_j \le t \le d_j - T$ **then**
        schedule $2y_j$ fraction of job j in calibration
      reset $y_j = 0$
    $carryover = 0$
    $C_t = C_t - frac \cdot C_t$
  $carryover = carryover + C_t$
  foreach job j, $y_j = y_j + X_{jt}$

that time. Figure 2 shows an example of this process. The resulting calibrations are assigned to machines in round-robin fashion. We use $\mathcal{C}$ to denote the schedule of calibrations produced by the rounding step. Our third and final step is to assign jobs to calibrations, given by Algorithm 2. First we double the calibration schedule $\mathcal{C}$ using twice as many machines (for $6m'$ in total). Then we can scan the calibrations in increasing time order, and assign jobs using earliest-deadline-first scheduling. More precisely, we choose a job with earliest deadline from those unscheduled jobs obeying the TISE constraint, with ties broken arbitrarily. If there is still room in the calibration, schedule the job. Otherwise, finish this calibration and move on to the next one.

Note that the rounding step discards any of the job assignments X_{jt}, so it should not be obvious that our algorithm schedules all jobs. We shall show that a fractional assignment of jobs to calibrations is still feasible after the rounding step. Intuitively, this assignment shows that a preemptive schedule is possible on those calibrations. Our final EDF step transforms the preemptive schedule into a nonpreemptive one. This transformation does not work in general with EDF scheduling, so it should not be obvious that it works here.

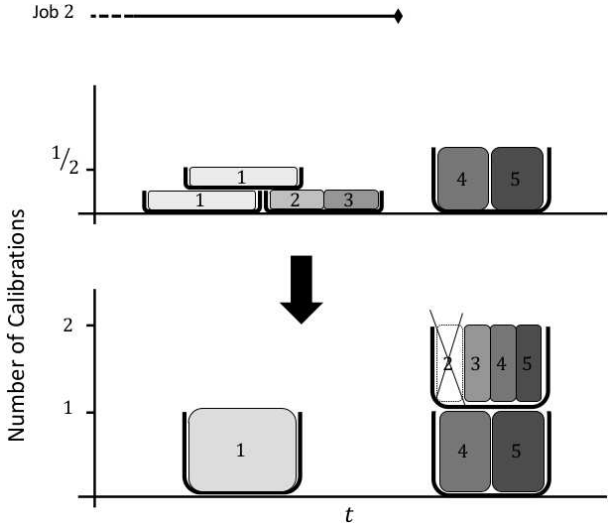

Figure 3: Example of the fractional job assignment generated by Algorithm 3 on the calibration schedule shown in Figure 2. Buckets represent calibrations, and shaded rectangles represent job assignments. (The ordering of jobs within a bucket does not matter, but the width indicates the amount of the calibration consumed by the job assignment.) In this example, the rounding process delays job 2, whose window is indicated by the line at the top, past its deadline. Consequently, this assignment is discarded; the idea of Corollary 6 is to show that such discarding can only occur if the job is already sufficiently scheduled.

Correctness and performance analysis

We begin by focusing on Algorithm 1. For correctness, we must show two things. First, we show that the rounded calibrations are valid (i.e., calibrations do not overlap on a machine). Second, we show that the calibration schedule has a feasible, fractional job assignment. Finally, we conclude that the rounding step has only a constant-factor overhead.

LEMMA 4. *The rounding process (Algorithm 1) produces a valid schedule $\mathcal{C}$ of calibrations on $3m'$ machines, where no two calibrations on the same machine overlap.*

PROOF. Due to constraint 1, slightly restated, the LP solution guarantees that for all t, we have $\sum_{t' \in \mathcal{T}, t \le t' < t+T} C_t \le m'$. That is, there are at most m (total fractional) calibrations started in any time-T period. The rounding process delays at most a $1/2$ calibration at a time. It follows that the number of integer calibrations output in the range $[t, t+T)$ can be at most $\lfloor 2 \cdot (1/2 + \sum_{t' \in \mathcal{T}, t \le t' < t+T} C_t) \rfloor \le 2(1/2 + m') = 2m' + 1 \le 3m'$

Because we have at most $3m'$ integer calibrations started within a size-T window, the oldest calibration must end before the $(3m'+1)$th calibration begins. Thus, round-robin scheduling suffices. □

We next prove that the rounded calibration schedule permits a fractional assignment of all jobs. We do so constructively by using an augmented algorithm, Algorithm 3, that maps job assignments while creating rounded calibrations. Because this is an existential proof, the modified algorithm is only used for the proof. The idea of Algorithm 3 is similar to Algorithm 1. Namely, continue to accrue fractional calibrations *carryover* until reaching a total of exactly $1/2$ a calibration, then create a new full calibration. Logically, the fractional calibrations (and all their jobs) are simply delayed. The

augmentation is that when accruing the fractional calibrations, also record the fractions y_j of jobs being delayed, and write them into the full calibration whenever they obey the TISE constraint. Figure 3 gives an example of this process. Because some of the job can be delayed past a TISE-feasible calibration, we overschedule a $2y_j$ fraction of the job.

LEMMA 5. *At any point when executing Algorithm 3, we have $y_j \le carryover$. That is, the unscheduled fraction of job j being delayed is at most the unscheduled fraction of calibration being delayed. Moreover, the fraction of jobs fits within the fractional calibration, i.e., $\sum_j y_j p_j \le carryover \cdot T$.*

PROOF. The proof is by induction over iterations. From the pseudocode, y_j only increases by $frac \cdot X_{jt}$ after *carryover* increases by $frac \cdot C_t$. Similarly, at the end of the loop, y_j only increases by X_{jt} after *carryover* increases by C_t. From LP constraint (2), $C_t \ge X_{jt}$, so *carryover* increases by at least as much y_j. Using a similar argument with LP constraint (3), we also have $C_t T \ge \sum_j X_{jt} p_j$, so the increases to $y_j p_j$ cannot exceed the increases to $carrover \cdot T$ □

COROLLARY 6. *For every job j, the fractional assignments of job j to calibrations produced by Algorithm 3 sum to at least 1. Moreover, the total work assigned to any calibration does not exceed T.*

PROOF. Consider a time t when deciding whether to schedule a $2y_j$ fraction of job j, i.e., determining whether the current calibration obeys the TISE constraint for the job. If $X_{jt'} > 0$ for any $t' \ge t$, then the current calibration is feasible, because the LP only assigns jobs to feasible calibrations. Thus, the only time the calibration can be infeasible for job j is the last time y_j is reset. By Lemma 5, $y_j \le carryover \le 1/2$ at this point. Thus, at least the first half of the job is scheduled. The extra factor of 2 when scheduling the job means that the job is (at least) fully scheduled. The bound on total work follows from Lemma 5 even if the $2y_j$ fraction of the job is always scheduled in the calibration. □

LEMMA 7. *Assuming the TISE instance is feasible on m' machines, the calibration schedule $\mathcal{C}$ resulting from Algorithm 1 is a feasible set of calibrations such that all jobs can be (fractionally) assigned to calibrations without violating the TISE constraint. Moreover, $\mathcal{C}$ uses at most $3m'$ machines and $2C^*$ calibrations, where C^* is the minimum possible number of calibrations on m' machines.*

PROOF. The correctness of the calibrations and feasibility of the fractional assignment follow from Lemma 4 and Corollary 6. Moreover, $\mathcal{C}$ also uses $3m'$ machines by construction.

To get the bound on calibrations, observe that any feasible TISE schedule is feasible for the LP. Thus, the optimal solution to the LP has value at most C^*. The rounding process doubles the number of calibrations (creating a full calibration for every $1/2$ calibration produced by the LP), yielding a total of at most $2C^*$ calibrations. □

We next turn to our earliest-deadline-first (EDF) variant (Algorithm 2) scheduling the jobs within the calibrations. We interpret an assignment of jobs to calibrations $\mathcal{C}$ as a preemptive schedule. We first show that whenever a preemptive schedule is possible on $\mathcal{C}$, a preemptive version of EDF is feasible. Perhaps unsurprisingly, this proof is similar to the preemptive optimality of EDF when considering the classic problem without calibrations. Then we show that for a TISE instance, EDF can be transformed into a nonpreemptive schedule by doubling the number of machines.

For the purposes of the proof only, we define the ***fractional EDF*** algorithm as follows (similar to Algorithm 2). Consider the calibrations in nondecreasing order of start time. For the current calibration, let J' be the set of unscheduled fractional jobs whose windows contain the calibration (i.e., obeying the TISE constraint). Let $j \in J'$ be the job with the earliest deadline, with ties broken by job number. Assign as much of job j as possible to the calibration. When J' is empty or the calibration is full, continue to the next calibration.

LEMMA 8. *Consider a valid integer calibration schedule $\mathcal{C}$. Suppose that a fractional TISE assignment of jobs to calibrations is feasible. Then the fractional EDF strategy produces a feasible fractional TISE job assignment to calibration schedule $\mathcal{C}$.*

PROOF. Suppose for the sake of contradiction that the EDF schedule is not feasible. Let S be a feasible TISE schedule that shares the longest possible prefix with the EDF schedule. Consider the earliest point at which S and EDF differ, and let k be the rank of the calibration (in sorted order) during which that difference occurs. Then S schedules some job j during calibration k, whereas EDF schedules a different job j'. Because this is the first point of difference, S must schedule j' to some later calibration k'. Swap (as much as possible) of j with j'. If this swap is feasible, this contradicts the assumption that there is a longest matching prefix.

We must prove that this swap is feasible. Because j is not the EDF job, we know $d_j \ge d_{j'}$. Moreover, j' is the EDF job, so it must have release time before the start of calibration k and hence also before calibration k'. It follows that calibration k' is feasible for job j with respect to the TISE constraint. □

LEMMA 9. *Suppose that EDF produces a valid fractional TISE assignment on calibration schedule $\mathcal{C}$. Then there is a feasible integer schedule using twice as many machines and calibrations, specifically by duplicating $\mathcal{C}$ across another set of machines.*

PROOF. Let S be the fractional EDF schedule on $\mathcal{C}$. Consider each calibration in turn. If the last job assigned to that calibration is fractional, instead assign the full job to the corresponding mirrored calibration. Remove any other fractional pieces of the job.

Because at most one new fractional job can be created at the end of each calibration, this process resolves all fractional assignments. □

The following lemma states that Algorithm 2 is at least as good as the mapping in the preceding lemma. In some sense, this lemma is not necessary—we could instead use the algorithm of Lemma 9 in place of Algorithm 2. But we think Algorithm 2 is more natural, so we opt for a small increase to the analytical complexity.

LEMMA 10. *After the kth calibration, all jobs completed by the fractional EDF transformation of Lemma 9 are also completed by Algorithm 2.*

PROOF. The proof is by induction on k. For each job chosen by fractional EDF, either it has the earliest deadline and hence our algorithm would also choose it, or the job has already been executed. □

Finally, we conclude by giving bounds on the full TISE algorithm as well as the ISE algorithm.

LEMMA 11. *Assuming the TISE instance is feasible on m' machines using at most C^* calibrations, our TISE algorithm produces a feasible schedule on $6m'$ machine using at most $4C^*$ calibrations.*

PROOF. From Lemma 7, the LP and calibration-rounding steps produces a feasible calibration schedule using at most $2C^*$ calibrations on $3m'$ machines. Lemma 10 states that our algorithm is at least as good as the fractional-EDF-to-integer transformation of Lemma 9. Combining this fact with Lemma 8, we conclude that our algorithm produces a feasible (integer) schedule using twice the number of machines and calibrations, i.e., $6m'$ machines and $4C^*$ calibrations. □

The following theorem states that using our TISE algorithm to solve an ISE instance gives an $O(1)$ approximation using $O(1)$ machine augmentation and no speed augmentation.

THEOREM 12. *Consider any feasible long-job ISE instance on m machines. Let C^* denote the minimum possible number of calibrations to feasibly solve the problem on m machines. Then running our TISE algorithm on that instance produces a feasible TISE schedule using at most $18m$ machines and $12C^*$ calibrations. Moreover, a TISE schedule is also a valid ISE schedule.*

PROOF. This follows directly from Combining the factor of 3 from Lemma 2 with Lemma 11. □

Trading speed augmentation for machine augmentation

Thus far we have shown how to construct an $O(1)$-machine 1-speed $O(1)$-approximation for the ISE problem (producing a more restricted TISE solution). We conclude this section by showing how to transform this TISE solution into a 1-machine $O(1)$-speed $O(1)$-approximation. The fact that we are working with long jobs and a TISE solution is pivotal here. It is not clear how to make this sort of transformation in general.

Suppose we have a TISE schedule on cm machines, for integer c. (Following from Theorem 12, we shall set $c = 18$.) Here is the algorithm. Group the machines arbitrarily into groups of c machines that will all map to one target speed-$2c$ machine. First, construct the calibration schedule on the target machine as follows. Start at time $t = 0$. Repeat the following steps. If any calibration on the source machine includes timestep t, calibrate the target machine at time t, advance to time $t = t + T$, and repeat. Otherwise, increase t to the next earliest calibration on any of the source machines. This calibration schedule guarantees that every calibrated timestep on any of the source machines is also a calibrated timestep on the target machine.

Next, consider each calibration interval $[t, t+T)$ on the target machine in any order. For any calibration ℓ on a source machine $i \in \{0, 1, \ldots, c-1\}$ that fully contains the first half of the target calibration, i.e., $[t, t+T/2)$, assign ℓ to the size-$T/(2c)$ time interval $[t+iT/2c, t+(i+1)T/2c)$. Keep the jobs in the same order within the interval; just scale the processing times by a factor of $1/2c$. Perform a similar process for each source calibration fulling containing the second half, $[t+T/2, T)$, of the target calibration.

LEMMA 13. *Given a TISE schedule on cm speed-1 machines with C calibrations, the above algorithm produces an ISE schedule on m speed-2c machines with at most C calibrations.*

PROOF. Consider a single group of c source machines. At most one source machine is mapped to each size-$T/(2c)$ subinterval of a target calibration. Moreover, because calibrations on each machine do not overlap, at most one calibration on each machine can be mapped there.

These mappings are feasible because a) the target interval is fully contained in the source calibration, and b) we start from a TISE instance, meaning that the jobs can be run at any time within their calibration.

To see that every source calibration is mapped somewhere, suppose that a source calibration only overlaps part of the end of $[t, t+T)$. Then there is another calibration on the target machine at time $[t+T, t+2T)$. The source calibration must either overlap half of $[t, t+T)$ or half of $[t+T, T+2T)$. A similar argument applies on the front end of target calibrations. Thus, every source calibration is mapped.

Finally, to count the number of calibrations, consider the calibration process. A calibration only occurs on the target machines if 1) there is a calibration on some source machine at the same time, or 2) there is a calibration on some source machine between the previous calibration and the current one. Thus, we can charge all target calibrations against source calibrations. □

Combining this speed transformation with Theorem 12, we get the following theorem, meaning a 1-machine $O(1)$-speed $O(1)$-approximation.

THEOREM 14. *Consider any feasible long-job ISE instance on m machines. Let C^* denote the minimum possible number of calibrations to feasibly solve the problem on m machines. Then running our TISE algorithm followed by the machine-to-speed transformation produces a feasible ISE schedule using at most m machines, each at 36 speed, with at most $12C^*$ calibrations.* □

4. STRATEGY FOR SHORT-WINDOW JOBS

This section presents an ISE algorithm for the special case that all jobs have short windows. Our algorithm exploits similarities between the ISE problem and the classic machine minimization (MM) problem, applying any MM algorithm as a black box while asymptotically preserving its approximation guarantees.

Our main idea stems from the following simplified case. Suppose that all jobs fall within a single size-T time interval, i.e., $\max_j \{d_j\} - \min_j \{r_j\} \leq T$. Then an optimal ISE solution uses either 0 or 1 calibration per machine. Thus, minimizing the number of machines and minimizing the number of calibrations are equivalent, and applying the MM algorithm as a black box yields a good solution to the ISE instance. With some manipulation, we generalize this relationship and use it to construct solutions for a short-window ISE input. Note that applying the MM algorithm globally does not work, because there may be long periods of time where we can use fewer machines (and hence fewer calibrations).

Throughout this section, to avoid confusion about where constants are being introduced we use γT to denote the maximum window length of a short job, i.e., $\gamma = 2$ according to Definition 1.

The algorithm

The rough idea of our algorithm is as follows. Partition time into size-$2\gamma T$ intervals. For each interval, consider the subset of jobs whose windows are contained inside the interval. Apply an MM algorithm to the interval. Transform the MM schedule to an ISE schedule by adding appropriate calibrations, but executing all jobs at the same time as before. The final schedule is the union of the schedules for each interval.

The simple partitioning strategy does not quite work because there may be arbitrarily many jobs whose windows span the boundaries separating intervals. Fortunately, there is a trivial fix for this issue: partition time again but at an offset of γT, and schedule any remaining jobs on a new set of machines as before. This revised partitioning step is given as pseudocode in Algorithm 4. For clarity, the partitioning pseudocode as presented is linear in the length of the schedule, but it is straightforward to transform the code to be polynomial in the number of jobs.

Algorithm 4: Partitioning short jobs into length-$2\gamma T$ intervals

Let J_{short} be the set of all short-window jobs
Allocate disjoint sets of machines M_1 and M_2

$t \leftarrow 0$
while $t \leq \max_j \{d_j\}$ **do**
  Let $J' \subseteq J_{short}$ be the jobs nested in $[t, t+2\gamma T)$,
    i.e., with $t \leq r_j < d_j \leq t+2\gamma T$.
  Schedule J' on machines M_1 using Algorithm 5
  $t \leftarrow t+2\gamma T$

Remove from J_{short} any jobs scheduled above
$t \leftarrow \gamma T$
while $t \leq \max_j \{d_j\}$ **do**
  Let $J' \subset J_{short}$ be the jobs nested in $[t, t+2\gamma T)$,
    i.e., with $t \leq r_j < d_j \leq t+2\gamma T$.
  Schedule J' on machines M_2 using Algorithm 5
  $t \leftarrow t+2\gamma T$

Algorithm 5: Scheduling each length-$2\gamma T$ interval

Let t be the start time of the interval
Let J' be the set of jobs assigned to this interval

Run an MM algorithm on J' to produce schedule S
Let w be the number of machines used by S

Use $3w$ machines for S'
for $i = 1$ *to* w **do**
  calibrate machine i at $t+kT$ for $k \in \{0, 1, 2, \ldots, 2\gamma - 1\}$
for *each job* $j \in J'$ **do**
  Let m_j be the machine to which j is assigned in S
  Let x_j be the start time of j in S
  if *j is not a crossing job* **then**
    assign j to machine m_j at time x_j in S'
  else
    if *j is a k-th crossing job for even k* **then**
      calibrate machine $w+m_j$ at time x_j in S'
      assign j to machines $w+m_j$ at time x_j in S'
    else
      calibrate machine $2w+m_j$ at time x_j in S'
      assign j to machine $2w+m_j$ at time x_j in S'

For each of the intervals $[t, t+2\gamma T)$ produced by the partitioning step, we produce a subschedule for the jobs J' in that interval as follows (see Algorithm 5). First construct an MM schedule S for jobs J' using w machines. Note that the MM schedule S specifies a start time x_j for each job as well as a machine on which to run that job, but an ISE schedule S' must also specify a schedule of calibrations on each machine. Moreover, S' must ensure that each job's execution fall fully within a single calibration. We transform the MM schedule S to ISE schedule S' as follows. First calibrate each of the w machines 2γ times. Next, map S to S' by preserving the times at which the jobs are executed. The remaining question is how to assign jobs to machines. There are two cases. If a job is fully contained in a calibration, it is assigned to the same machine in S' as in S. The more challenging case arises when a job crosses calibration boundaries, an issue that we cope with by introducing more machines.[3] We call a job j a ***k-th crossing job*** if the start time x_j of the job falls in the k-th calibration, i.e., $t+kT \leq x_j < t+(k+1)T$, but the completion time of the job falls in a different calibration, i.e., $x_j + p_j > t+(k+1)T$. For each machine used by S, introduce a new machine to handle crossing jobs for odd k, and another machine to handle crossing jobs for even k. For each crossing job, we create a new calibration dedicated specifically to the job. As we shall prove, all of the jobs in S' fall fully within a calibration, and no calibrations on a single machine overlap, so the schedule S' is a feasible schedule.

Correctness and performance analysis

To see that the algorithm produces a valid schedule, we first show that the subschedules produced for each interval are feasible. We then show that the main algorithm combines these interval schedules without introducing any conflicts.

LEMMA 15. *Consider any set of short jobs J' with windows nested inside a time interval $[t, t+2\gamma T)$. Algorithm 5 produces a valid ISE schedule for these jobs.*

PROOF. A feasible MM schedule has two main properties. 1) Every job is scheduled nonpreemptively within its window, i.e., starting no earlier than its release time and finishing no later than its deadline. 2) Jobs on the same machine cannot have overlapping execution periods. An ISE schedule adds two additional restrictions, namely: 3) Every job's execution must be contained fully within a calibration on the machine to which it is assigned, and 4) for each machine, the calibrations on that machine must be nonoverlapping.

Algorithm 5 starts with a feasible MM schedule, preserving all execution times, so property (1) holds trivially. Moreover, for each machine in the MM schedule, the jobs assigned to that machine are spread across three machines in the ISE schedule, so (2) also holds trivially.

We next show properties (3) and (4). For the first w machines, where w is the number of machines used by the MM schedule, property (4) holds by construction—calibrations are performed exactly every T timesteps. Moreover, noncrossing jobs satisfy property (3) on those machines by definition. For each crossing job, Algorithm 5 creates a new calibration, thereby satisfying property (3). We need only argue that those calibrations do not overlap each other, and hence the schedule also observes property (4). Consider two crossing jobs assigned to the same machine in the ISE schedule. Because they are assigned to the same machine, they must have the same crossing parity (even or odd). Moreover, they must start from the same machine in the MM schedule, so those crossing numbers must differ by at least 2. Thus, the scheduled start times for the jobs must differ by at least T, meaning that the calibrations do not overlap. □

LEMMA 16. *Our short-window algorithm (combining Algorithms 4 and 5) produces a valid schedule for an ISE instance of short-window jobs.*

PROOF. We first argue that all jobs are assigned to some interval in the partitioning step, and hence all jobs are part of some interval schedule. We then argue that the interval schedules do not interfere, and hence a final schedule can be formed by taking the union across intervals. Combining these two facts implies that the overall schedule is feasible.

Consider a particular job j in the partitioning step. If j is scheduled during the first loop of Algorithm 4, we are done. Suppose

[3] If a calibration is allowed to be performed before the previous calibration ends, then no extra machines are necessary, just extra calibrations. We focus here on the more difficult version of the ISE problem, where calibrations cannot be invoked less than T timesteps of each other.

instead that j is not scheduled during the first loop. Then j's window crosses a multiple of $2\gamma T$, say $2k\gamma T$ for integer k. Because j is short, its window has length at most γT, and hence $r_j \geq 2k\gamma T - \gamma T$ and $d_j \leq 2k\gamma T + \gamma T$. Thus, j's window is contained completely in the interval $[(2k-1)\gamma T, (2k+1)\gamma T)$, and j is assigned to an interval in the second loop.

We next argue that the interval schedule in Algorithm 5 only creates calibrations nested inside the interval $[t, 2\gamma T)$. The calibrations for noncrossing jobs fall inside $[t, 2\gamma T)$ by construction as long as γ is an integer. Crossing jobs, on the other hand, must have start times within the range $[t, (2\gamma - 1)T)$, so those calibrations are also inside the interval. Thus, taking the union of interval schedules is feasible as long as the intervals themselves are disjoint (which is true for the first or second loop of Algorithm 4). □

Next we analyze the performance of our short-window algorithm. This analysis has a few components. First, we argue that for each interval instance, the MM solution serves as a lower bound for the number of calibrations for the ISE problem. Then we extend the lower bound across a set of disjoint intervals as produced by each pass of the partitioning phase. Finally, we conclude by arguing that our algorithm only loses a constant factor beyond the MM algorithm applied.

LEMMA 17. *Consider any set of short jobs J' with windows nested inside a time interval $[t, t + 2\gamma T)$. Let w^* be the value of the optimal solution to the MM problem on J', i.e., the minimum number of machines. Then the ISE problem requires at least w^* calibrations and machines.*

PROOF. The lemma follows from the fact that all feasible solutions require at least w^* machines, and each machine must be calibrated at least once to schedule any jobs. □

LEMMA 18. *For fixed offset time τ, suppose J_i is a set of short jobs with windows nested inside time interval $[\tau + 2i\gamma T, \tau + 2(i+1)\gamma T)$. Let w_i^* be the value of the optimal solution to the MM problem on J_i.*

Then any feasible solution to the ISE problem on the $r+1$ disjoint intervals $\cup_{i=0}^{r} J_i$ requires at least $\max_i w_i^*$ machines. Moreover, an optimal solution to the ISE problem on $\cup_{i=0}^{r} J_i$ requires at least $\sum_{i=0}^{r} w_i^*/2$ calibrations.

PROOF. From Lemma 17, each subset requires w_i^* machines. Adding more jobs only increases the number of machines. Thus $\max_i w_i^*$ is a lower bound.

Consider every other interval $J_0, J_2, J_4, \ldots$ or $J_1, J_3, J_5, \ldots$. Intervals J_i and J_{i+2} are separated by much more than T timesteps. So no calibration used for J_i can also be used for J_{i+2}. Thus, following from Lemma 17, $w_0^* + w_2^* + w_4^* + \cdots$ is a lower bound for the minimum possible number of calibrations. Similarly, $w_1^* + w_3^* + w_5^* + \cdots$ is also a lower bound. Taking the maximum of the two, we have a lower bound of $\sum_{i=0}^{r} w_i^*/2$ calibrations. □

LEMMA 19. *Consider any set of short jobs J' with windows nested inside a time interval $[t, t + 2\gamma T)$. Let w be the number of machines found by the black-box MM algorithm. Then our ISE solution on J' performs at most $4\gamma w$ calibrations on $3w$ machines.*

PROOF. Consider a single machine in the MM solution. Our algorithm constructs three machines for the ISE schedule. The first of these machines gets calibrated every T timesteps, for 2γ calibrations. Each of the crossing jobs is assigned to one of the other two machines, with one calibration per crossing job. Since there can be at most $2\gamma - 1$ crossing jobs, there are at most $4\gamma - 1$ calibrations arising from this machine. Multiplying by the number w of machines completes the proof. □

Finally, we conclude that for constant γ, our algorithm asymptotically preserves the approximation guarantees of the MM algorithm applied.

THEOREM 20. *Consider any short-job instance J_{short}. Let w^* denote the minimum possible number of machines among feasible ISE schedules. Let C^* be the minimum possible number of calibrations among feasible ISE schedule.*

Suppose that we have a black-box α-approximation algorithm to the MM problem. Then our ISE algorithm produces a feasible ISE schedule on at most $6\alpha w^* = O(\alpha w^*)$ machines using at most $16\gamma\alpha C^* = O(\alpha C^*)$ calibrations.

PROOF. Consider a sequence of disjoint intervals as defined in Lemma 18. From Lemma 18, we have $C^* \geq \sum_i w_i^*/2$ and $w^* \geq \max_i w_i^*$, where w_i^* is the optimal number of machines for the ith interval. For the ith interval, the MM algorithm finds a solution using at most αw_i^* machines. Thus applying Lemma 19, Algorithm 5 makes at most $(4\gamma)\alpha w_i^*$ calibrations on $3w_i^*$ machines. Summing across all i, our ISE algorithm uses at most $4\gamma\alpha \sum_i w_i^* \leq 8\gamma\alpha C^*$ calibrations on $3\max_i \alpha w_i^* \leq 3\alpha w^*$ machines.

We lose a factor of 2 in both bounds because Algorithm 4 runs on two sets of disjoint intervals using disjoint machines. □

5. OTHER RELATED WORK

Beyond its practical applications, ISE is an interesting interval scheduling variation because it is often optimal to delay the scheduling of a job. This property is unusual in more standard metrics like machine minimization and throughput maximization. However, it is certainly not unique.

Interval scheduling for power minimization is a popular [1] and ostensibly similar problem when the goal can be reduced to minimizing idle periods for a continuous interval schedule. Like ISE, this makes starting work on a machine expensive, which tends to discourage scheduling a job as early as possible and reward job clustering. It should be noted however that since calibrations last a discrete amount of time, the problems are subtly different. Baptiste et al. [3, 4] give an $O(n^5)$-time dynamic programming based algorithm for finding an optimal solution on a single processor with preemption, reducing to $O(n^4)$-time in the unit-job case. Demaine et al. [9] extend their work to a multi-processor environment, yielding a polynomial-time optimal algorithm for the unit-job case.

Chang et al. [7] propose a model for minimizing active processor time which is deceptively similar to ISE. Instead of calibrations length T, they consider timesteps of depth B. That is, up to B jobs can be scheduled in the same timestep at no additional cost and the goal is to minimize the active number of time-steps. However, they consider only preemptive scheduling and don't offer an approximation for the $B > 2$, NP-complete version.

6. CONCLUSIONS

In this paper, we showed how to reduce the ISE problem to the MM problem, producing approximation guarantees that are almost as good as those for the MM problem. To within constant factors, we have also argued that this is the best possible. In the case that all jobs have long windows, our algorithm is asymptotically optimal.

Because the best general approximation to the MM problem is an $O(\sqrt{\log n / \log\log n})$ approximation, we made little effort to minimize the constants in this paper. We think that some of the constants in the reduction could be reduced. That said, partitioning jobs into long and short jobs inherently has an overhead of at least 2 in terms of both machines and calibrations. It would be nice to achieve constants that look like $(1+\varepsilon)$, but that would require some new ideas.

Acknowledgments

This research was supported in part by National Science Foundation grants CCF-1218188 and CCF-1314633.

7. REFERENCES

[1] S. Albers. Energy-efficient algorithms. *Commun. ACM*, 53(5):86–96, May 2010.

[2] N. Bansal, H.-L. Chan, R. Khandekar, K. Pruhs, B. Schieber, and C. Stein. Non-preemptive min-sum scheduling with resource augmentation. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 614–624, Oct 2007.

[3] P. Baptiste. Scheduling unit tasks to minimize the number of idle periods: A polynomial time algorithm for offline dynamic power management. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 364–367, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics.

[4] P. Baptiste, M. Chrobak, and C. Dürr. Polynomial-time algorithms for minimum energy scheduling. *ACM Trans. Algorithms*, 8(3):26:1–26:29, July 2012.

[5] M. A. Bender, D. P. Bunde, V. J. Leung, S. McCauley, and C. A. Phillips. Efficient scheduling to minimize calibrations. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 280–287, New York, NY, USA, 2013.

[6] C. Burroughs. New integrated stockpile evalution program to better ensure weapons stockpile safety, security, reliability. `http://www.sandia.gov/LabNews/060331.html`, march 2006.

[7] J. Chang, H. Gabow, and S. Khuller. A model for minimizing active processor time. In L. Epstein and P. Ferragina, editors, *Algorithms - ESA 2012*, volume 7501, pages 289–300. Springer Berlin Heidelberg, 2012.

[8] J. Chuzhoy, S. Guha, S. Khanna, and J. Naor. Machine minimization for scheduling jobs with interval constraints. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 81–90, Oct 2004.

[9] E. D. Demaine, M. Ghodsi, M. T. Hajiaghayi, A. S. Sayedi-Roshkhar, and M. Zadimoghaddam. Scheduling to minimize gaps and power consumption. In *Proceedings of the 19th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 46–54, New York, NY, USA, 2007.

[10] R. Graham, E. Lawler, J. Lenstra, and A. Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Disc. Math.*, 5:287–326, 1979.

[11] S. Im, S. Li, B. Moseley, and E. Torng. A dynamic programming framework for non-preemptive scheduling problems on multiple machines. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 1070–1086, 2015.

[12] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, July 2000.

[13] C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation (extended abstract). In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, pages 140–149, New York, NY, USA, 1997.

[14] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, Dec. 1987.