A Unified Approach to Scheduling on Unrelated Parallel Machines*

V. S. Anil Kumar† Madhav V. Marathe‡ Srinivasan Parthasarathy§
Aravind Srinivasan¶

Abstract

We develop a single rounding algorithm for scheduling on unrelated parallel machines; this algorithm works well with the known linear programming-, quadratic programming-, and convex programming-relaxations for scheduling to minimize completion time, makespan, and other well-studied objective functions. This algorithm leads to the following applications for the general setting of unrelated parallel machines: (i) a bicriteria algorithm for a schedule whose weighted completion-time and makespan *simultaneously* exhibit the current-best individual approximations for these criteria; (ii) better-than-two approximation guarantees for scheduling to minimize the L_p norm of the vector of machine-loads, for all $1 < p < \infty$; and (iii) the first constant-factor multicriteria approximation algorithms that can handle the weighted completion-time and *any* given collection of integer L_p norms. Our algorithm has a natural interpretation as a melding of linear-algebraic and probabilistic approaches. Via this view, it yields a common generalization of rounding theorems due to Karp *et al.* and Shmoys & Tardos, and leads to improved approximation algorithms for the problem of scheduling with resource-dependent processing times introduced by Grigoriev *et al.*

1 Introduction

The complexity and approximability of scheduling problems for multiple machines is an area of active research [17, 20]. A particularly general (and challenging) case involves scheduling on *unrelated parallel machines*, where the processing times of jobs depend arbitrarily on the machines to which they are assigned. That is, we are given n jobs and m machines, and each job needs to be scheduled on exactly one machine; we are also given a collection of integer values $p_{i,j}$ such that if we schedule job j on machine i, then the processing time of operation j is $p_{i,j}$. Three major objective functions, all NP-hard, in this context are to minimize the weighted completion-time of the jobs, the L_p norm of the loads on the machines, and the maximum completion-time of the machines, or the *makespan* (i.e., the L_∞ norm of the machine-loads) [18, 21, 22, 4]. There is no single measure that is considered "universally good", and therefore there has been much interest in *simultaneously* optimizing many given objective functions: if there is a schedule that simultaneously has cost T_i with respect to objective i for each i, we aim to efficiently construct a schedule that has cost $\lambda_i T_i$ for the ith objective, for each i. (One typical goal here is to keep all the λ_i small.) The

*A preliminary version of this paper appeared as the paper "Approximation Algorithms for Scheduling on Multiple Machines", in the *Proc. IEEE Symposium on Foundations of Computer Science*, pages 254–263, 2005.

†Department of Computer Science, Virginia Tech, Blacksburg 24061. Email: akumar@vbi.vt.edu

‡Virginia Bio-informatics Institute and Department of Computer Science, Virginia Tech, Blacksburg 24061. Email: mmarathe@vbi.vt.edu

§IBM T. J. Watson Research Center, 19, Skyline Drive, Hawthorne, NY 10532. Work done while at the Department of Computer Science, University of Maryland, College Park, MD 20742. Email: spartha@us.ibm.com

¶Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Email: srin@cs.umd.edu

current-best approximation algorithms for these measures are very much tailored to the individual measure. We develop a unified approach to all of these problems, leading to better approximation algorithms for the single-criterion and multi-criteria versions.

We will primarily focus on approximation algorithms, since all problems considered herein are NP-hard. Most of the current approaches for these single-criterion or multi-criteria problems are based on constructing fractional solutions by different linear programming (LP)-, quadratic programming-, and convex programming-relaxations and then rounding them into integral solutions. Two major rounding approaches for these problems are those of Lenstra, Shmoys & Tardos and Shmoys & Tardos [18, 21], and classical randomized rounding (Raghavan & Thompson [19]) as applied to specific problems by Skutella [22] and Azar & Epstein [4]. We develop a *single* rounding technique that works with all of these relaxations, gives improved bounds for scheduling under the L_p norms, and most importantly, helps develop schedules that are good for multiple combinations of the completion-time and L_p-norm criteria. For the case of simultaneous weighted completion time and makespan objectives, our approach yields a bicriteria approximation with the best-known guarantees for both these objectives. We start by presenting four of our applications, and then discuss our rounding technique and other implications thereof.

(i) **Simultaneous approximation of weighted completion-time and makespan.** In the weighted completion-time objective problem, we are given an integral weight w_j for each job; we need to assign each job to a machine, and also order the jobs assigned to each machine, in order to minimize the weighted completion-times of the jobs. The current-best approximations for weighted completion-time and makespan are 3/2 [22] and 2 [18], respectively. We construct schedules that achieve these bounds *simultaneously*: if there exists a schedule with (weighted completion-time, makespan) $\leq (C, T)$ coordinate-wise, our schedule has a pair $\leq (1.5C, 2T)$. This is noticeably better than the bounds obtained by using general bicriteria results for (weighted completion-time, makespan) such as Stein & Wein [24] and Aslam, Rasala, Stein & Young [2]: e.g., we would get $\leq (2.7C, 3.6T)$ using the methods of [24]. More importantly, note that if we can improve one component of our pair $(1.5, 2)$ (while worsening the other arbitrarily), we would improve upon the current-best approximation known for weighted completion-time or makespan.

(ii) **Minimizing the L_p norm of machine loads.** Note that the makespan is the L_∞ norm of the machine loads, and that the L_1 norm is easily minimizable. The L_p norms of the machine loads, for $1 < p < \infty$, interpolate between these "minmax" and "minsum" criteria. See, e.g., [5] for an example that motivates the L_2 norm. A breakthrough of Azar & Epstein [4] improves upon the $\Theta(p)$-approximation for minimizing the L_p norm of machine loads [3], by presenting a 2-approximation for each $p > 1$, and a $\sqrt{2}$-approximation for $p = 2$. Our algorithm further improves upon [4] by giving better-than-2 approximation algorithms for all p, $1 \leq p < \infty$: e.g., we get approximations of 1.585, $\sqrt{2}$, 1.381, 1.372, 1.382, 1.389, 1.41, and 1.436 for $p = 1.5$, 2, 2.5, 3, 3.5, 4, 4.5, and 5 respectively.

(iii) **Multicriteria approximations for completion time and multiple L_p norms.** There has been much interest in schedules that are simultaneously near-optimal w.r.t. multiple objectives and in particular, multiple L_p norms [7, 1, 5, 6, 9, 14] in various special cases of unrelated parallel machines. For unrelated parallel machines, it is easy to show instances where, for example, any schedule that is reasonably close to optimal w.r.t. the L_2 norm will be far from optimal for, say, the L_∞ norm; thus, such simultaneous approximations cannot hold. However, we can still ask multi-criteria questions. Given an arbitrary (finite, but not necessarily of bounded size) set of positive integers $p_1, p_2, \ldots, p_r$, suppose we are given that there exists a schedule in which: (a) for each i, the L_{p_i} norm of the machine loads is at most some given T_i, and (b) the weighted completion-time is at most some given C. We show how to efficiently construct a schedule in which the L_{p_i} norm of the machine loads is at most $3.2 \cdot T_i$ for each i, and the weighted completion-time is at most $3.2 \cdot C$. To our knowledge, this is the first such multi-criteria approximation algorithm with a constant-factor approximation guarantee. We also present several additional results, some of which generalize our application (i) above, and others that improve upon the results of [5, 9].

(iv) **Convergence to fairness.** All the above applications apply to "one-shot" problems. Many of our results have certain additional properties that lead to quick convergence to *fairness* for all machines with high probability, when *multiple* scheduling problems need to be solved on a set M of m machines. One such consequence is as follows. Suppose our goal is makespan minimization, and that we use our randomized algorithm (called SchedRound) on a sequence of scheduling problems (with possibly different sets of jobs) on the set M of m machines. Let i denote some machine, and k be an index of one of these scheduling problems. Let $L_{i,k}$ be the random variable denoting the total load on machine i in problem k, and let OPT_k be the optimal makespan for problem k. Normalize to define $Z_{i,k} = L_{i,k}/OPT_k$. $Z_{i,k}$ is a cost metric which we want to keep small, as close to 1 as possible. We guarantee that $Z_{i,k} \leq 2$ with probability one; however, our approach helps us show the following for multiple executions. Define $\overline{Z}_i(N)$ to be the average of the $Z_{i,k}$ values for $k = 1, 2, \ldots, N$. We can show that if $N \geq K(\log m)/\epsilon^2$ for a certain absolute constant K, then with high probability, we have simultaneously for all machines i that $\overline{Z}_i(N) \leq (1 + \epsilon)$. That is, in the "repeated executions" setting, we converge quickly – in $O(\log m)$ executions whose inputs can be chosen adversarially – to being fair on all machines with high probability, with no knowledge of future inputs being necessary. Thus, even for objectives such as makespan minimization for which we do not improve upon the current-best approximation guarantee (which is two [18]), we get such an improvement in the "multiple executions" setting; we are not aware of other methods that achieve this.

Our approach in brief. Once again, all of the above applications follow by applying our rounding approach in combination with some problem-specific ideas. We now provide a sketch of SchedRound, our rounding algorithm. Suppose we are given a fractional assignment $\{x^*_{i,j}\}$ of jobs j to machines i; i.e., $\sum_i x^*_{i,j} = 1$ for all j, with all the $x^*_{i,j}$ being non-negative. Let $t^*_i = \sum_j p_{i,j} x^*_{i,j}$ be the fractional load on machine i. We round the $x_{i,j}$ in iterations by a melding of linear algebra and randomization. Let $X^{(h)}_{i,j}$ denote the random value of $x_{i,j}$ at the end of iteration h. For one, we maintain the invariant that $\mathbf{E}[X^{(h)}_{i,j}] = x^*_{i,j}$ for all i, j and h. Second, we "protect" each machine i almost until the end: the load $\sum_j p_{i,j} X^{(h)}_{i,j}$ on i at the end of iteration h equals its initial value t^*_i with probability 1, until the remaining fractional assignment on i falls into a small set of simple configurations. Informally, these two properties respectively capture some of the utility of independent randomized rounding [19] and those of [18, 21]. Importantly, while SchedRound is fundamentally based on linear systems, we show in Lemma 3.1 that it has good behavior w.r.t. a certain family of *quadratic* functions as well. Similarly, the precise details of our rounding help us show better-than-2 approximations for L_p norms of the machine-loads.

We then interpret SchedRound in a general linear-algebraic setting, and show that it yields further applications. A basic result of Karp et al. [13], shows that if $A \in \Re^{m \times n}$ is a "column-sparse" matrix, then for any given real vector $x = (x_1, x_2, \ldots, x_n)^T$, we can efficiently find a *rounded* counterpart $X = (X_1, X_2, \ldots, X_n)^T$ such that $\|AX - Ax\|_\infty$ is "small". The generalization of our rounding approach achieves the same bound on $\|AX - Ax\|_\infty$ with probability 1, with the additional property that for all j, $\mathbf{E}[X_j] = x_j$. This yields the result of [21]; furthermore, we use this generalization to obtain new multicriteria approximations in the setting of [13].

Thus, SchedRound helps improve upon various basic results in scheduling. In particular, different rounding techniques have thus far been applied for diverse objective functions: e.g., the approach of [18, 21] in [4] for general L_p norms, and independent randomized rounding [19] for weighted completion time in [22] and for the special case of the L_2 norm in [4]. SchedRound unifies and strengthens all these results. Furthermore, since it works with differing objective functions such as weighted completion-time and L_p norms of machine loads, it is the first approximation algorithm to construct schedules that are good w.r.t. many such objectives simultaneously. We thus expect our approach to be of use in further contexts as well.

Our main algorithm SchedRound, and its linear-algebraic generalization LinAlgRand ("Linear Algebra and Randomization"), are presented in Section 2. The applications to approximation algorithms are

presented in Sections 3, 4 and 5. The applications of the linear-algebraic generalization are then developed in Section 6. We discuss fairness in Section 7, and conclude in Section 8. We provide certain routine proof-details in the Appendix.

2 The Main Rounding Algorithm, and a Generalization

We start by presenting algorithm SchedRand in Section 2.1. We then observe in Section 2.2 that there is a natural interpretation of this algorithm as a melding of linear algebra and randomization for general linear systems. This interpretation will lead to further applications, in Section 6.

We now describe two notions that will be useful. Call a linear system $Ax = b$ (with A, x, b given) *canonical* if $x_j \in (0, 1)$ for all j. Next, suppose a given linear system $Ax = b$ is *canonical and under-determined*. Then, $\texttt{RandStep}(A, x, b)$ is a randomized procedure that returns an updated version of x as follows. Since the linear system is under-determined, we can efficiently find a nonzero vector r such that $Ar = 0$. Since x is canonical, we can also efficiently find the strictly-positive quantities α and β such that: (i) all entries of $(x + \alpha \cdot r)$ and $(x - \beta \cdot r)$ lie in $[0, 1]$, and (ii) at least one entry of each of $(x + \alpha \cdot r)$ and $(x - \beta \cdot r)$ lies in $\{0, 1\}$. Then,

> with probability $\frac{\beta}{\alpha+\beta}$, $\texttt{RandStep}(A, x, b)$ returns the vector $x + \alpha \cdot r$;
> with the complementary probability of $\frac{\alpha}{\alpha+\beta}$, it returns $x - \beta \cdot r$.

Thus, $\texttt{RandStep}(A, x, b)$ yields a random vector Y which satisfies

$$\Pr[AY = b] = 1, \tag{1}$$

and in which at least one entry has been rounded. It is also easy to check that

$$\forall j,\ \mathbf{E}[Y_j] = x_j. \tag{2}$$

Note: Given a linear system $\mathcal{L}$ where it will be cumbersome to explicitly write out the matrix A etc., we may also refer to $\texttt{RandStep}(A, x, b)$ as $\texttt{RandStep}(\mathcal{L}, x)$. `RandStep` will be used in essentially all of our algorithms.

2.1 The basic algorithm SchedRound

Algorithm SchedRound takes as input a fractional assignment x^* of jobs to machines, as well as the processing time $p_{i,j}$ of each job j on each machine i, and produces an integral assignment. We start with some background and notation in this paragraph. Let $x^*_{i,j} \in [0, 1]$ denote the fraction of job j assigned to machine i in x^*, and note that for all j, $\sum_i x^*_{i,j} = 1$. Initialize $x = x^*$. The algorithm iteratively modifies x such that x becomes integral in the end. At least one coordinate of x is rounded to zero or one during each iteration; we will throughout *maintain the invariant* "$\forall j,\ \sum_i x_{i,j} = 1$". Once a co-ordinate is rounded to 0 or 1, it is *unchanged* from then on. The algorithm is composed of **Phase 1** followed by **Phase 2**, each composed of some number of iterations. The (random) values at the *end* of iteration h of the overall algorithm, will be denoted $X^{(h)}_{i,j}$. Each iteration h will conduct a randomized update to the fractional assignment that will guarantee the invariant that for all (i, j), $\mathbf{E}[X^{(h)}_{i,j}] = x^*_{i,j}$. Furthermore, Phase I will attempt to "protect" the machines "as much as possible": for each machine i, we will try to maintain its load at the end of iteration h, $\sum_j p_{i,j} X^{(h)}_{i,j}$, equal to its initial value $t^*_i = \sum_j p_{i,j} x^*_{i,j}$ for as long as possible (i.e., for all h up to as large value as possible). Once we move on to Phase 2, the yet-to-be

rounded variables will induce a simple structure, and in Phase 2, we will in fact not even consider the values $p_{i,j}$. We now present the details of the algorithm.

Notation. Let M denote the set of machines and J denote the set of jobs; let $m = |M|$ and $n = |J|$. As mentioned above, the (random) values at the *end* of iteration h will be denoted $X_{i,j}^{(h)}$.

SchedRound will first go through Phase 1, followed by Phase 2 (one of these phases could be empty). We start by saying when we transition from Phase 1 to Phase 2, and then describe a generic iteration in each of these phases. Suppose we are at the *beginning* of some iteration $h+1$ of the overall algorithm; so, we are currently looking at the values $X_{i,j}^{(h)}$. Let a job j be called a *floating job* if it is currently assigned fractionally to more than one machine. Let a machine i be called a *floating machine* if it currently has at least one floating job assigned to it. Machine i is called a *singleton machine* if it has **exactly one** floating job assigned to it currently. Let J' and M' denote the current set of floating jobs and **non-singleton** floating machines respectively. Let $n' = |J'|$ and $m' = |M'|$. Define V to be the set of yet-unrounded pairs currently; i.e., $V = \{(i,j) : X_{i,j}^{(h)} \in (0,1)\}$, and let $v = |V|$. We emphasize that all these definitions are w.r.t. the values at the beginning of iteration $(h+1)$. The current iteration (the $(h+1)^{st}$ iteration) is a Phase 1 iteration if $v > m' + n'$; at the first time we observe that $v \leq m' + n'$, we move to Phase 2. So, we initially have some number of iterations at the start of each of which, we have $v > m' + n'$; these constitute Phase 1. Phase 2 starts at the beginning of the first iteration where we have $v \leq m' + n'$. We next describe iteration $(h+1)$, based on which phase it is in.

Case I: *Iteration* $(h+1)$ *is in Phase 1.* Let J', M', n', m', V and v be as defined in the previous paragraph, and recall that $v > m' + n'$. Consider the following linear system:

$$\forall j \in J', \ \sum_{i \in M} x_{i,j} = 1; \tag{3}$$

$$\forall i \in M', \ \sum_{j \in J'} x_{i,j} \cdot p_{i,j} = \sum_{j \in J'} X_{i,j}^{(h)} \cdot p_{i,j}. \tag{4}$$

(**Remark:** It is important to note that index i is allowed to take any value in M in the sum in (3), but that the universal quantification for i in (4) is only over M'.) The point $P = (X_{i,j}^{(h)} : \ i \in M, j \in J')$ is a feasible solution for the variables $\{x_{i,j}\}$, and all the coordinates of P lie in $(0,1)$. Crucially, the number of variables v in the linear system $\mathcal{L}$ given by (3) and (4) exceeds the number of constraints $n' + m'$. We now obtain $X^{(h+1)}$ by running `RandStep`$(\mathcal{L}, P)$. (Note that the components of $X^{(h)}$ that lie outside of V are unchanged.) Now, (1) shows that $X^{(h+1)}$ still satisfies (3) and (4); we have rounded at least one further variable, and also have $\mathbf{E}[X_{i,j}^{(h+1)}] = X_{i,j}^{(h)}$ for all i, j, by (2).

Case II: *Iteration* $(h+1)$ *is in Phase 2.* Let J', M' etc. be defined w.r.t. the values at the start of this (i.e., the $(h+1)^{st}$) iteration. Consider the bipartite graph $G = (M, J', E)$ in which we have an edge (i,j) between job $j \in J'$ and machine $i \in M$ iff $X_{i,j}^{(h)} \in (0,1)$. We employ the bipartite dependent-rounding algorithm of Gandhi *et al.* [8]. Choose an even cycle $\mathcal{C}$ or a *maximal* path $\mathcal{P}$ in G, and partition the edges in $\mathcal{C}$ or $\mathcal{P}$ into two matchings $\mathcal{M}_1$ and $\mathcal{M}_2$ (it is easy to see that such a partition exists and is unique). Define positive scalars α and β as follows.

$$\alpha = \min\{\kappa > 0 : \ ((\exists (i,j) \in \mathcal{M}_1 : X_{i,j}^{(h)} + \kappa = 1) \bigvee (\exists (i,j) \in \mathcal{M}_2 : X_{i,j}^{(h)} - \kappa = 0))\};$$

$$\beta = \min\{\kappa > 0 : \ ((\exists (i,j) \in \mathcal{M}_1 : X_{i,j}^{(h)} - \kappa = 0) \bigvee (\exists (i,j) \in \mathcal{M}_2 : X_{i,j}^{(h)} + \kappa = 1))\}.$$

(Note that these definitions appear similar to those of `RandStep`. We will examine this issue, as well as the reason why we do not use the values $p_{i,j}$ in Case II, in Section 2.2.) We execute the following randomized step, which rounds at least one variable to 0 or 1:

With probability $\beta/(\alpha+\beta)$, set
$X_{i,j}^{(h+1)} := X_{i,j}^{(h)} + \alpha$ for all $(i,j) \in \mathcal{M}_1$, and $X_{i,j}^{(h+1)} := X_{i,j}^{(h)} - \alpha$ for all $(i,j) \in \mathcal{M}_2$;
with the complementary probability of $\alpha/(\alpha+\beta)$, set
$X_{i,j}^{(h+1)} := X_{i,j}^{(h)} - \beta$ for all $(i,j) \in \mathcal{M}_1$, and $X_{i,j}^{(h+1)} := X_{i,j}^{(h)} + \beta$ for all $(i,j) \in \mathcal{M}_2$.

This completes the description of a typical iteration of Phase 2. Hence, it also completes our algorithm-description.

Note that the algorithm requires at most mn iterations, since at least one further variable gets rounded in each iteration. We next present some useful observations and results about the algorithm.

Define machine i to be *protected* during iteration $h+1$ if iteration $h+1$ was in Phase 1, and if i was **not** a singleton machine at the start of iteration $h+1$. If i was then a non-singleton floating machine, then since Phase 1 respects (4), we will have, for any given value of $X^{(h)}$, that

$$\sum_{j \in J} X_{i,j}^{(h+1)} \cdot p_{i,j} = \sum_{j \in J} X_{i,j}^{(h)} \cdot p_{i,j} \tag{5}$$

with probability one. This of course also holds if i had no floating jobs assigned to it at the beginning of iteration $h+1$. Thus, if i is protected in iteration $(h+1)$, the total (fractional) load on it is the same at the beginning and end of this iteration with probability 1.

Lemma 2.1 *(i) In any iteration of Phase 2, any floating machine has at most two floating jobs assigned fractionally to it. (ii) Let ϕ and J' denote the fractional assignment and set of floating jobs respectively, at the beginning of Phase 2. For any values of these random variables, we have with probability one that for all $i \in M$, $\sum_{j \in J'} X_{i,j} \in \{\lfloor \sum_{j \in J'} \phi_{i,j} \rfloor, \lceil \sum_{j \in J'} \phi_{i,j} \rceil\}$, where X denotes the final rounded vector.*

Proof: We start by making some observations about the beginning of the first iteration of Phase 2. Consider the values v, m', n' at the beginning of that iteration. At this point, we had $v \leq n' + m'$; also observe that $v \geq 2n'$ and $v \geq 2m'$ since every job $j \in J'$ is fractionally assigned to at least two machines and every machine $i \in M'$ is a non-singleton floating machine. Therefore, we must have $v = 2n' = 2m'$; in particular, we have that every non-singleton floating machine has *exactly two floating jobs fractionally assigned to it.* The remaining machines of interest, the singleton floating machines, have exactly one floating job assigned to them. This proves part (i).

Recall that each iteration of Phase 2 chooses a cycle or a *maximal* path. So, it is easy to see that if i had two fractional jobs j_1 and j_2 assigned fractionally to it at the beginning of iteration $h+1$ in Phase 2, then we have $X_{i,j_1}^{(h+1)} + X_{i,j_2}^{(h+1)} = X_{i,j_1}^{(h)} + X_{i,j_2}^{(h)}$ with probability 1. This equality, combined with part (i), helps us prove part (ii). □

The following useful lemma is a simple exercise for the reader:

Lemma 2.2 *For all i, j, h, u, $\mathbf{E}[X_{i,j}^{(h+1)} \mid (X_{i,j}^{(h)} = u)] = u$. In particular, $\mathbf{E}[X_{i,j}^{(h)}] = x_{i,j}^*$ for all i, j, h.*

Lemma 2.3 *(i) Let machine i be protected during iteration $h+1$. Then $\forall h' \leq h+1$, $\sum_{j \in J} X_{i,j}^{(h')} \cdot p_{i,j} = \sum_{j \in J} x_{i,j}^* \cdot p_{i,j}$ with probability 1. Let X denote the final rounded vector. (ii) For all i, $\sum_{j \in J} X_{i,j} \cdot p_{i,j} < \sum_{j \in J} x_{i,j}^* \cdot p_{i,j} + \max_{j \in J:\ X_{i,j}=1} p_{i,j}$ with probability 1. (iii) For all i, $\sum_{j \in J} X_{i,j} \cdot p_{i,j} < \sum_{j \in J} x_{i,j}^* \cdot p_{i,j} + \max_{j \in J:\ x_{i,j}^* \in (0,1)} p_{i,j}$ with probability 1.*

Proof: Part (i) follows from (5), and from the fact that if a machine was protected in any one iteration, it is also protected in all previous ones.

We now argue part (ii). If i remained protected throughout the algorithm, then its total load never changes and the lemma holds. Let $h_{unp}(i)$ denote the first iteration at which machine i became unprotected. Let $J_{unp}(i)$ denote the set of *floating* jobs at the start of iteration $h_{unp}(i)$ which were assigned to machine i at the end of the algorithm. There are four possible cases. **Case (a):** Machine i became a singleton machine when it became unprotected. If case (a) does not occur, then i had two floating jobs j_1 and j_2 when it became unprotected (Lemma 2.1(i) shows that this is the only other possibility); let the fractional assignments of j_1 and j_2 on i at this time be ϕ_{i,j_1} and ϕ_{i,j_2} respectively. **Case (b):** $\phi_{i,j_1} + \phi_{i,j_2} \in (0,1]$. **Case (c):** $\phi_{i,j_1} + \phi_{i,j_2} \in (1,2]$, and strictly one of the jobs j_1 and j_2 belongs to J_{unp}. **Case (d):** $\phi_{i,j_1} + \phi_{i,j_2} \in (1,2]$, and both j_1 and j_2 belongs to J_{unp}. The total load on machine i when it became unprotected is $\sum_{j \in J} x^*_{i,j} \cdot p_{i,j}$. Hence, in cases (a), (b), and (c), the additional load on machine i at the end of the algorithm is strictly less than $\max_{j \in J_{unp}} p_{i,j}$. We now consider case (d); in this case, the additional load on i is $(1-\phi_{i,j_1})p_{i,j_1} + (1-\phi_{i,j_2})p_{i,j_2} \leq (2-\phi_{i,j_1}-\phi_{i,j_2}) \cdot \max_{j_1,j_2}\{p_{i,j_1}, p_{i,j_2}\} < 1 \cdot \max_{j \in J_{unp}(i)} p_{i,j}$. The strict inequality follows due to the fact that $\phi_{i,j_1}+\phi_{i,j_2} > 1$ in case (d). Since $\max_{j \in J_{unp}(i)} p_{i,j} \leq \max_{j \in J: X_{i,j}=1} p_{i,j}$, part (ii) of the lemma holds.

We now argue part (iii). From the proof of part (ii), it follows that the final load on machine i is strictly less than $\sum_{j \in J} x^*_{i,j} \cdot p_{i,j} + \max_{j \in J_{unp}(i)} p_{i,j}$ with probability 1. Job j belongs to $J_{unp}(i)$ only if $x^*_{i,j} \in (0,1)$; hence, with probability 1 the final load on machine i is strictly less than $\sum_{j \in J} x^*_{i,j} \cdot p_{i,j} + \max_{j \in J: x^*_{i,j} \in (0,1)} p_{i,j}$. This concludes the proof of the lemma. □

Algorithm SchedRound underlies almost all of the algorithms discussed further in this paper, and hence we will employ the above lemmas in various applications below.

2.2 A linear-algebraic interpretation

We now observe that SchedRound can be interpreted more generally as follows. Suppose we have a linear system $Ax = b$, with A, b, and x given. We wish to round x to some integral X such that each X_j is the ceiling or floor of x_j, and so that AX "approximately" equals b. We will present and analyze a partially-specified algorithm LinAlgRand for this task. We then see how SchedRound is essentially an instantiation of LinAlgRand, with the caveat that we may change the linear system as we pass from Phase I to Phase II of SchedRound. Section 6 will exploit the fact that LinAlgRand works with general linear systems, in order to develop further algorithmic applications.

Given a linear system $A'x' = b'$ where $x'_j \in [0,1]$ for all j, we define an operation `Simplify`(A', x', b') which modifies (A', x', b') as follows. Let $S = \{j : x'_j \in \{0,1\}\}$. Modify (A', x', b') by removing the columns corresponding to S and entries corresponding to S from A' and x' respectively, and replacing each b'_i by $b'_i - \sum_{j \in S} A'_{i,j} x'_j$. Note that this leads to an equivalent but canonical linear system. It also ensures that once rounded to 0 or 1, a variable x_j never changes value from then on.

Given a linear system $Ax = b$, consider the following (partially-specified) rounding algorithm LinAlgRand:

Algorithm LinAlgRand:
{**Comment:** By subtracting out integer parts, we assume that $x_j \in [0,1]$ for all j.}

Initialize $A' \leftarrow A$, $x' \leftarrow x$, and $b' \leftarrow b$;
`Simplify`(A', x', b');
While there exists some variable to be rounded in x' do:
(**Comment:** $A'x' = b'$ is the current *canonical* linear system.)

"Judiciously" remove some constraints from the system $A'x' = b'$ so that it becomes under-determined;
$x' \leftarrow$ `RandStep`(A', x', b');
`Simplify`(A', x', b').

End of Algorithm LinAlgRand

The partially-unspecified part of the algorithm is which rows to eliminate in a "judicious fashion" in each iteration. In Section 6, we will study an approach of Karp *et al.* [13] for such row-elimination for certain families of linear constraints; we will employ LinAlgRand along with this approach to generalize the results of [13].

The following lemma summarizes some useful properties of LinAlgRand:

Lemma 2.4 *Given an initial system $Ax = b$, suppose algorithm LinAlgRand rounds x to some X, using some rule for choosing the rows to be eliminated in each round. Let n be the number of components of x. We have the following: (i) $\forall j$, $X_j \in \{\lfloor x_j \rfloor, \lceil x_j \rceil\}$ with probability 1, and the algorithm terminates within n iterations; (ii) $\forall j$, $\mathbf{E}[X_j] = x_j$, and (iii) if a certain constraint of the original system $Ax = b$ was not removed until the end of iteration h, then that constraint holds with probability one for the (random) n-dimensional vector X that we have at the iteration h.*

Proof: Part (i) is straightforward. Part (ii) follows by repeated application of (2) and Bayes' theorem. Finally, part (iii) follows from (1). □

Connection to Algorithm SchedRound. Let us now see why algorithm SchedRound is a special case of LinAlgRand. It is easily seen that the randomized update of Case I of SchedRound, where we maintain (3) and (4), is an instantiation of LinAlgRand. Although we do not "judiciously remove any constraints" here, we have implicitly done so by neglecting the constraints (4) for singleton machines.

Next, suppose we are in iteration $(h+1)$, which is in Case II of SchedRound. Consider the bipartite graph $G = (M, J', E)$ as described in Case II; given an edge $e = (i, j)$ of this graph, let $X_e^{(h)}$ denote $X_{i,j}^{(h)}$. Given any vertex (job or machine) v of G, let $N(v)$ denote the set of *edges* incident on v at the end of iteration h, and let $s(v) = \sum_{e \in N(v)} X_e^{(h)}$. The linear system to which LinAlgRand is basically being applied to in iteration $(h+1)$ of SchedRound, is:

$$\forall v, \quad \sum_{e \in N(v)} x_e = s(v). \tag{6}$$

Starting with the solution $x_e = X_e^{(h)}$ for all edges e, we can see that iteration $(h+1)$ of SchedRound proceeds as follows. If it found an even cycle $\mathcal{C}$ in G, it considers the restriction of (6) to the nodes v contained in $\mathcal{C}$. This system is under-determined already, and the randomized update of Case II is as prescribed by LinAlgRand. (The fact that this system is under-determined is one reason why we drop consideration of the processing times $p_{i,j}$ in Phase 2 of SchedRound.) If a maximal path $\mathcal{P}$ was found instead, we consider the restriction of (6) to the nodes v contained in $\mathcal{P}$. This system is not under-determined, and Case II basically proceeds by implicitly dropping the constraints of (6) that correspond to the two endpoints v of $\mathcal{P}$. Letting ℓ be the number of vertices in $\mathcal{P}$, this leads to a system with $\ell - 1$ variables and $\ell - 2$ constraints, and is hence under-determined; the update `RandStep` is then applied.

Thus, SchedRound is essentially a special case of LinAlgRand; however, we change the linear system when we pass from Phase I to Phase II. We will see further applications of LinAlgRand in Section 6.

3 Weighted Completion Time and Makespan

We now use algorithm SchedRound to develop a $(\frac{3}{2}, 2)$-bicriteria approximation algorithm for (weighted completion time, makespan) with unrelated parallel machines. That is, given a pair (C, T), where C is the target value of the weighted completion time and T, the target makespan, our algorithm either proves that no schedule exists which simultaneously satisfies both these bounds, or yields a solution whose cost is at most $(\frac{3C}{2}, 2T)$. Our algorithm builds on the quadratic-programming formulation of Skutella [22] and some key properties of SchedRound; as we will see, the makespan bound needs less work, but managing the weighted completion time simultaneously needs much more care. Let w_j denote the weight of job j. For a given assignment of jobs to machines, the sequencing of the assigned jobs can be done optimally on each machine i by applying Smith's ratio rule [23]: schedule the jobs in the order of non-increasing ratios $\frac{w_j}{p_{i,j}}$. Let this order on machine i be denoted $\prec_i$. Let x be an "assignment-vector" as before: i.e., $\sum_i x_{i,j} = 1$ for all jobs j, with all the $x_{i,j}$ being non-negative. For each machine i, define a potential function

$$\Phi_i(x) = \sum_{(k,j):\ k \prec_i j} w_j x_{i,j} x_{i,k} p_{i,k}.$$

Note that if x is an *integral* assignment, then $\sum_i \sum_{k:\ k \prec_i j} x_{i,j} x_{i,k} p_{i,k}$ is the amount of time that job j waits before getting scheduled. Thus, for integral assignments x, the total weighted completion time is

$$(\sum_{i,j} w_j p_{i,j} x_{i,j}) + (\sum_i \Phi_i(x)). \tag{7}$$

Given a pair (C, T), we write the following Integer Quadratic Program (IQP) motivated by [22]. The $x_{i,j}$ are the usual assignment variables, and z denotes an upper bound on the weighted completion time. The IQP is to minimize z subject to "$\forall j,\ \sum_i x_{i,j} = 1$", "$\forall i, j,\ x_{i,j} \in \{0, 1\}$", and:

$$z \geq (\sum_j w_j \sum_i \frac{x_{i,j}(1 + x_{i,j})}{2} p_{i,j}) + (\sum_i \Phi_i(x)); \tag{8}$$

$$z \geq \sum_j w_j \sum_i x_{i,j} p_{i,j}; \tag{9}$$

$$\forall i,\ T \geq \sum_j p_{i,j} x_{i,j}; \tag{10}$$

$$\forall (i,j),\ (p_{i,j} > T) \Rightarrow (x_{i,j} = 0). \tag{11}$$

The constraint (11) is easily seen to be valid, since we want solutions of makespan at most T. Next, since $d(1 + d)/2 = d$ for $d \in \{0, 1\}$, (7) shows that constraints (8) and (9) are valid: z denotes an upper bound on the weighted completion time, subject to the makespan being at most T. Crucially, as shown in [22], the quadratic constraint (8) is *convex*, and hence the convex-programming relaxation (CPR) of the IQP wherein we set $x_{i,j} \in [0, 1]$ for all i, j, is solvable in polynomial time. Technically, we can only solve the relaxation to within an additional error ϵ that is, say, any positive constant. As shown in [22], this is easily dealt with by derandomizing the algorithm by using the method of conditional probabilities. Let ϵ be a suitably small positive constant. We find a (near-)optimal solution to the CPR, with additive error at most ϵ. If this solution has value more than $C + \epsilon$, then we have shown that (C, T) is an infeasible pair. Else, we construct an integral solution by running SchedRound the fractional assignment x. Assuming that we obtained such a fractional assignment, let us now analyze this algorithm.

Recall that $X^{(h)}$ denotes the (random) fractional assignment at the end of iteration h of SchedRound. We next present a lemma that claims the key property that for each machine i, the expected potential function value $\mathbf{E}[\Phi_i(X^{(h)})]$ is non-increasing as a function of h; we prove the lemma using the structure of SchedRound.

Lemma 3.1 *For all i and h, $\mathbf{E}[\Phi_i(X^{(h+1)})] \leq \mathbf{E}[\Phi_i(X^{(h)})]$.*

Proof: Fix a machine i and iteration h. Let us condition on the event that the fractional assignment at the end of iteration h, $X^{(h)}$ equals some arbitrary but fixed $x^{(h)} = \{x_{i,j}^{(h)}\}$. We will now show that, conditioning on this event, $\mathbf{E}[\Phi_i(X^{(h+1)})] \leq \Phi_i(x^{(h)})$. We may assume that $\Phi_i(x^{(h)}) > 0$, since $\mathbf{E}[\Phi_i(X^{(h+1)})] = 0$ if $\Phi_i(x^{(h)}) = 0$. We first show by a perturbation argument that the value

$$\zeta = \frac{\mathbf{E}[\Phi_i(X^{(h+1)})]}{\Phi_i(x^{(h)})}$$

is maximized when all jobs with nonzero weight have the same $\frac{w_j}{p_{i,j}}$ ratio. Partition the jobs into sets $S_1, \ldots, S_k$ such that in each partition, the jobs have the same $\frac{w_j}{p_{i,j}}$ ratio. Let the ratio for set S_g be r_g and let $r_1, \ldots, r_k$ be in non-decreasing order. For each job $j \in S_1$, we set $w'_j = w_j + \lambda p_{i,j}$ where λ has sufficiently small absolute value so that the relative ordering of $r_1, \ldots, r_k$ does not change. This changes the value of ζ to a new value $\zeta'(\lambda) = \frac{a+b\lambda}{c+d\lambda}$, where a, b, c and d are values independent of λ, $\zeta = a/c$, and $a, c > 0$. Crucially, since $\zeta'(\lambda)$ is a ratio of two linear functions, its value depends monotonically (either increasing or decreasing) on λ, in the allowed range for λ. Hence, there exists an allowed value for λ such that $\zeta'(\lambda) \geq \zeta$, and either r'_1 (which is $r_1 + \lambda$) equals r_2, or $r'_1 = 0$. The terms for jobs with zero weight can be removed. We continue this process until all jobs with non-zero weight have the same ratio $\frac{w_j}{p_{i,j}}$. So, we assume w.l.o.g. that all jobs have the same value of this ratio; thus we can rewrite, for some fixed value $\xi > 0$,

$$\begin{aligned}
\Phi_i(x^{(h)}) &= \xi \cdot \sum_{\{k,j\}:k \prec_i j} p_{i,j} p_{i,k} x_{i,j}^{(h)} x_{i,k}^{(h)}; \\
\mathbf{E}[\Phi_i(X^{(h+1)})] &= \xi \cdot \mathbf{E}\left[\sum_{\{k,j\}:k \prec_i j} p_{i,j} p_{i,k} X_{i,j}^{(h+1)} X_{i,k}^{(h+1)}\right].
\end{aligned}$$

(Again, the above expectations are taken conditional on $X^{(h)} = x^{(h)}$.) There are three possibilities for a machine i during iteration $h+1$:

Case I: *i is protected in iteration $h+1$.* In this case,

$$\begin{aligned}
\mathbf{E}[\Phi_i(X^{(h+1)})] &= \frac{\xi}{2} \cdot (\mathbf{E}[(\sum_j p_{i,j} X_{i,j}^{(h+1)})^2] - \sum_j \mathbf{E}[(p_{i,j} X_{i,j}^{(h+1)})^2]) \\
&= \frac{\xi}{2} \cdot ((\sum_j p_{i,j} x_{i,j}^{(h)})^2 - \sum_j \mathbf{E}[(p_{i,j} X_{i,j}^{(h+1)})^2]) \qquad (12)
\end{aligned}$$

where the latter equality follows since i is protected in iteration $h+1$. Further, for any j, the probabilistic rounding of Phase I of SchedRound ensures that there exists a pair of positive reals (α, β) such that $X_{i,j}(h+1)$ equals $(x_{i,j}^{(h)} + \alpha)$ with probability $\beta/(\alpha+\beta)$, and equals $(x_{i,j}^{(h)} - \beta)$ with the complementary probability. So,

$$\mathbf{E}[(X_{i,j}^{(h+1)})^2] = \frac{\beta}{\alpha+\beta} \cdot (x_{i,j}^{(h)} + \alpha)^2 + \frac{\alpha}{\alpha+\beta} \cdot (x_{i,j}^{(h)} - \beta)^2 \geq (x_{i,j}^{(h)})^2.$$

Plugging this into (12), we get that $\mathbf{E}[\Phi_i(X^{(h+1)})] \leq \Phi_i(x^{(h)})$ in this case.

Case II: *i is unprotected since it was a singleton machine at the start of iteration $h+1$.* Let j be the single floating job assigned to i. Then, $\Phi_i(X^{(h+1)})$ is a linear function of $X_{i,j}^{(h+1)}$, and so $\mathbf{E}[\Phi_i(X^{(h+1)})] = \Phi_i(x^{(h)})$ by Lemma 2.2 and the linearity of expectation.

Case III: *Iteration $h+1$ is in Phase 2, and i had two floating jobs then.* (Lemma 2.1(i) shows that this is the only remaining case.) Let j and j' be the floating jobs on i. $\Phi_i(X^{(h+1)})$ has: (i) constant terms, (ii) terms that are linear in $X_{i,j}^{(h+1)}$ or $X_{i,j'}^{(h+1)}$, and (iii) the term $X_{i,j}^{(h+1)} \cdot X_{i,j'}^{(h+1)}$ with a non-negative coefficient. Terms of type (i) and (ii) are handled by the linearity of expectation, just as in Case II. Now consider the term $X_{i,j}^{(h+1)} \cdot X_{i,j'}^{(h+1)}$; we claim that the two factors here are *negatively correlated*. Indeed, in each iteration of Phase 2, there are positive values α, β such that we set $(X_{i,j}^{(h+1)},\ X_{i,j'}^{(h+1)})$ to $(x_{i,j}^{(h)} + \beta,\ x_{i,j'}^{(h)} - \beta)$ with probability $\alpha/(\alpha+\beta)$, and to $(x_{i,j}^{(h)} - \alpha,\ x_{i,j'}^{(h)} + \alpha)$ with probability $\beta/(\alpha+\beta)$. Therefore,

$$\mathbf{E}[X_{i,j}^{(h+1)} \cdot X_{i,j'}^{(h+1)}] = (\alpha/(\alpha+\beta)) \cdot (x_{i,j}^{(h)} + \beta) \cdot (x_{i,j'}^{(h)} - \beta) + (\beta/(\alpha+\beta)) \cdot (x_{i,j}^{(h)} - \alpha) \cdot (x_{i,j'}^{(h)} + \alpha) \leq x_{i,j}^{(h)} \cdot x_{i,j'}^{(h)};$$

thus, the type (iii) term is also handled. □

Lemma 3.1 leads to our main theorem here.

Theorem 3.2 *Let C' and T' denote the total weighted completion time and makespan of the integral solution. Then, $E[C'] \leq (3/2) \cdot (C + \epsilon)$ for any desired constant $\epsilon > 0$, and $T' \leq 2T$ with probability 1; this can be derandomized to deterministically yield the pair $(3C/2, 2T)$.*

Proof: As shown in [22], the factor of ϵ can be easily disregarded by derandomizing the algorithm using the method of conditional probabilities. (We exploit the fact that all the values w_j and $p_{i,j}$ are integers, which implies that C is also an integer; thus, if the objective function is at most $(3/2) \cdot (C+\epsilon)$, then it must be at most $(3/2) \cdot C$ if $\epsilon < 1/3$.) The fact that $T' \leq 2T$ with probability 1 easily follows by applying Lemma 2.3(iii) with constraints (10) and (11). Let us now bound $\mathbf{E}[C']$.

Recall that $X = \{X_{i,j}\}$ denotes the final random integral assignment. Lemma 2.2 shows that $\mathbf{E}[X_{i,j}] = x_{i,j}^*$. Also, Lemma 3.1 shows that $\mathbf{E}[\Phi_i(X)] \leq \Phi_i(x^*)$, for all i. These, combined with the linearity of expectation, yields the following:

$$\mathbf{E}[(\sum_j w_j \sum_i p_{i,j} X_{i,j}/2) + (\sum_i \Phi_i(X))] \leq (\sum_j w_j \sum_i p_{i,j} x_{i,j}^*/2) + (\sum_i \Phi_i(x^*)) \leq z, \tag{13}$$

where the second inequality follows from (8). Similarly, we have

$$\mathbf{E}[\sum_j w_j \sum_i X_{i,j} p_{i,j}] = \sum_j w_j \sum_i x_{i,j}^* p_{i,j} \leq z, \tag{14}$$

where the inequality follows from (9). As in [22], we get from (7) that

$$\begin{aligned}
\mathbf{E}[C'] &= (\sum_{i,j} w_j p_{i,j} \mathbf{E}[X_{i,j}]) + (\sum_i \mathbf{E}[\Phi_i(X)]) \\
&= \mathbf{E}[(\sum_j w_j \sum_i p_{i,j} X_{i,j}/2) + (\sum_i \Phi_i(X))] + \mathbf{E}[\sum_j w_j \sum_i X_{i,j} p_{i,j}/2] \\
&\leq z + z/2 \\
&\leq 3C/2.
\end{aligned}$$

As mentioned at the beginning of this proof, we can derandomize this algorithm using the method of conditional probabilities. □

4 Minimizing the L_p Norm of Machine Loads

We now consider the problem of scheduling to minimize the L_p norm of the machine-loads, for some given $p > 1$. (The case $p = 1$ is trivial, and the case where $p < 1$ is not well-understood due to non-convexity.) We model this problem using a slightly different convex-programming formulation than Azar & Epstein [4]. Recall that J and M denote now the set of jobs and machines respectively. Let T be a target value for the L_p norm objective. Any feasible integral assignment with an L_p norm of at most T satisfies the following integer program (IP).

$$\forall j \in J \sum_i x_{i,j} \geq 1 \tag{15}$$

$$\forall i \in M \sum_j x_{i,j} \cdot p_{i,j} - t_i \leq 0 \tag{16}$$

$$\sum_i t_i^p \leq T^p \tag{17}$$

$$\sum_i \sum_j x_{i,j} \cdot p_{i,j}^p \leq T^p \tag{18}$$

$$\forall (i,j) \in M \times J \; x_{i,j} \in \{0,1\} \tag{19}$$

$$\forall (i,j) \in \{(i,j) \mid p_{i,j} > T\} \; x_{i,j} = 0 \tag{20}$$

We let $x_{i,j} \geq 0$ for all (i,j) in the above IP to obtain a convex program. The feasibility of the convex program can be checked in polynomial time to within an additive error of ϵ (for an arbitrary constant $\epsilon > 0$): the nonlinear constraint (17) is not problematic since it defines a convex feasible region [4]. We obtain the minimum feasible value of the L_p norm, T^*, using bisection search in the range $[\min_{i,j}\{p_{i,j}\}, \max_{i,j}\{p_{i,j}\}]$. We ignore the additive error ϵ in the rest of our discussions since our randomized guarantees can be converted into deterministic ones using the method of conditional probabilities in such a way that ϵ is eliminated from the final cost: the idea is the same as is sketched in the proof of Theorem 3.2. We also assume that T is set to T^* by a suitable bisection search. We round the fractional solution to the convex program, $\{x^*_{i,j}\}$, using SchedRound; we analyze the performance of the rounding below.

We start with the following two lemmas involving useful calculations; the proofs of these lemmas are presented in the Appendix.

Lemma 4.1 *Let $a \in [0,1]$ and $p, \lambda > 0$. Define $N(a,\lambda) = a\cdot(1+\lambda)^p + (1-a)$ and $D(a,\lambda) = (1+a\lambda)^p + a\lambda^p$. Let $\gamma(p) = \max_{(a,\lambda)\in[0,1]\times[0,\infty)} \frac{N(a,\lambda)}{D(a,\lambda)}$. Then, $\gamma(p)$ is at most: (i) 1, if $p \in (1,2]$; (ii) 2^{p-2}, if $p \in (2,\infty)$; and (iii) $O(2^p/\sqrt{p})$ if p is sufficiently large (i.e., there exist constants K and p_0 such that for all $p \geq p_0$, $\gamma(p) \leq K\cdot 2^p/\sqrt{p}$). Further, for $p = 2.5, 3, 3.5, 4, 4.5, 5, 5.5$ and 6, $\gamma(p)$ is at most $1.12, 1.29, 1.55, 1.86, 2.34, 3.05, 4.0$ and 5.36 respectively.*

Lemma 4.2 *Let a_1, a_2 be variables, each taking values in $[0,1]$. Let $D \doteq (\lambda_0 + a_1\cdot\lambda_1 + a_2\cdot\lambda_2)^p + a_1\lambda_1^p + a_2\lambda_2^p$, where $p > 1$, $\lambda_0 \geq 0$ and $\lambda_1, \lambda_2 > 0$ are arbitrary but fixed constants. Define N as follows:*
if $a_1 + a_2 \leq 1$, then $N = (1 - a_1 - a_2)\cdot\lambda_0^p + a_1\cdot(\lambda_0+\lambda_1)^p + a_2\cdot(\lambda_0+\lambda_2)^p$; else if $a_1 + a_2 \in (1,2]$, then $N = (1-a_2)\cdot(\lambda_0+\lambda_1)^p + (1-a_1)\cdot(\lambda_0+\lambda_2)^p + (a_1+a_2-1)\cdot(\lambda_0+\lambda_1+\lambda_2)^p$. Then, $N \leq \gamma(p)\cdot D$, where $\gamma(p)$ is as in Lemma 4.1.

To analyze the performance of SchedRound here, consider a fixed machine i. Recall that X denotes the final rounded assignment and $\{x^*_{i,j}\}$ the fractional solution to the convex program; let $t^*_i = \sum_j p_{i,j}x^*_{i,j}$

denote the load on i in the fractional solution. Let T_i denote the final (random) load on machine i. Let $U = \{U_{i,j}\}$ denote the random fractional assignment at the beginning of the first iteration during which i became unprotected. W.l.o.g., we assume that there are two distinct jobs j_1 and j_2 which are floating on machine i in assignment U. The cases where i became a singleton or i remains protected throughout the course of the algorithm are handled by setting one or both of the variables $\{U_{i,j_1}, U_{i,j_2}\}$ to zero; hence we do not consider these cases in the rest of our arguments.

The following simple lemma describes the joint distribution of (X_{i,j_1}, X_{i,j_2}), and will be useful in proving our main result here, Theorem 4.4.

Lemma 4.3 *Let u denote an arbitrary fractional assignment. Then the following holds.*

Case 1: *If $u_{i,j_1} + u_{i,j_2} \in [0,1]$, then*

$$\begin{aligned}
\Pr[((X_{i,j_1} = 1) \bigwedge (X_{i,j_2} = 1)) \mid U = u] &= 0 \\
\Pr[((X_{i,j_1} = 1) \bigwedge (X_{i,j_2} = 0)) \mid U = u] &= u_{i,j_1} \\
\Pr[((X_{i,j_1} = 0) \bigwedge (X_{i,j_2} = 1)) \mid U = u] &= u_{i,j_2} \\
\Pr[((X_{i,j_1} = 0) \bigwedge (X_{i,j_2} = 0)) \mid U = u] &= 1 - u_{i,j_1} - u_{i,j_2}
\end{aligned}$$

Case 2: *If $u_{i,j_1} + u_{i,j_2} \in (1,2]$, then*

$$\begin{aligned}
\Pr[((X_{i,j_1} = 1) \bigwedge (X_{i,j_2} = 1)) \mid U = u] &= u_{i,j_1} + u_{i,j_2} - 1 \\
\Pr[((X_{i,j_1} = 1) \bigwedge (X_{i,j_2} = 0)) \mid U = u] &= 1 - u_{i,j_2} \\
\Pr[((X_{i,j_1} = 0) \bigwedge (X_{i,j_2} = 1)) \mid U = u] &= 1 - u_{i,j_1} \\
\Pr[((X_{i,j_1} = 0) \bigwedge (X_{i,j_2} = 0)) \mid U = u] &= 0
\end{aligned}$$

Proof: If i never became unprotected, then both u_{i,j_1} and u_{i,j_2} are zero; we have Case 1 and the lemma holds trivially. If i became an unprotected singleton, then $u_{i,j_2} = 0$. Again, Case 1 occurs and the lemma can be easily seen to hold due to Lemma 2.2. Assume i become unprotected with both j_1 and j_2 fractionally assigned to it (i.e., $u_{i,j_1}, u_{i,j_2} \in (0,1)$). We now analyze Case 1. Since $u_{i,j_1} + u_{i,j_2} \in [0,1]$, it follows from Lemma 2.1 that $\Pr[((X_{i,j_1} = 1) \bigwedge (X_{i,j_2} = 1)) \mid U = u] = 0$. This implies that $\Pr[((X_{i,j_1} = 1) \bigwedge (X_{i,j_2} = 0)) \mid U = u] = \Pr[(X_{i,j_1} = 1) \mid U = u] = u_{i,j_1}$. The last equality follows from Lemma 2.2. By an identical argument, $\Pr[((X_{i,j_1} = 0) \bigwedge (X_{i,j_2} = 1)) \mid U = u] = u_{i,j_2}$. Finally, $\Pr[((X_{i,j_1} = 0) \bigwedge (X_{i,j_2} = 0)) \mid U = u]$ is the remaining value which is $1 - u_{i,j_1} - u_{i,j_2}$. We note that the above arguments hold because the events considered above are mutually exclusive and exhaustive. Case 2 is proved using very similar arguments. □

Theorem 4.4 *Given a fixed norm $p > 1$ and a fractional assignment whose fractional L_p norm is T, our algorithm produces an integral assignment whose value C_p satisfies $\mathbf{E}[C_p] \leq \rho(p) \cdot T$. Our algorithm can be derandomized in polynomial time to guarantee that $C_p \leq \rho(p) \cdot T$. The approximation factor $\rho(p)$ is at most the following: (i) $2^{\frac{1}{p}}$, for $p \in (1,2]$; (ii) $2^{1-1/p}$, for $p \in [2,\infty)$; and (iii) $2 - \Theta(\log p/p)$ for large p. For specific values $p > 2$, slightly better upper bounds for $\rho(p)$ can be computed using numerical techniques. In particular, the following table illustrates the achievable values of $\rho(p)$ for the corresponding values of p:*

p	2.5	3	3.5	4	p	4.5	5	5.5	6
$\rho(p)$	1.381	1.372	1.382	1.389	$\rho(p)$	1.410	1.436	1.460	1.485

Proof: Let $A(i) = \{j : U_{i,j} = 1\}$ and let $R_i = \sum_{j \in A(i)} p_{i,j}$ be the rounded load on i at the beginning of the first iteration in which i was unprotected. By definition of a protected machine, $R_i + U_{i,j_1} \cdot p_{i,j_1} + U_{i,j_2} \cdot p_{i,j_2} = t_i^*$. By Lemma 4.3, $\mathbf{E}[T_i^p \mid U = u]$ equals:

$$(1 - u_{i,j_1} - u_{i,j_2}) \cdot R_i^p + u_{i,j_1} \cdot (R_i + p_{i,j_1})^p + u_{i,j_2} \cdot (R_i + p_{i,j_2})^p \tag{21}$$

if $u_{i,j_1} + u_{i,j_2} \in [0, 1]$; and

$$(1 - u_{i,j_2}) \cdot (R_i + p_{i,j_1})^p + (1 - u_{i,j_1}) \cdot (R_i + p_{i,j_2})^p + (u_{i,j_1} + u_{i,j_2} - 1) \cdot (R_i + p_{i,j_1} + p_{i,j_2})^p \tag{22}$$

if $u_{i,j_1} + u_{i,j_2} \in (1, 2]$.

Let $\mu(x, i) = \sum_j x_{i,j} p_{i,j}^p$ for any assignment-vector x. Note that

$$t_i^{*p} + \mathbf{E}[\mu(X, i) \mid U = u] = t_i^{*p} + \sum_j u_{i,j} p_{i,j}^p \geq t_i^{*p} + u_{i,j_1} p_{i,j_1}^p + u_{i,j_2} p_{i,j_2}^p.$$

Combining this with the above-seen equality $R_i + U_{i,j_1} \cdot p_{i,j_1} + U_{i,j_2} \cdot p_{i,j_2} = t_i^*$, we get

$$t_i^{*p} + \mathbf{E}[\mu(X, i) \mid U = u] \geq (R_i + u_{i,j_1} \cdot p_{i,j_1} + u_{i,j_2} \cdot p_{i,j_2})^p + u_{i,j_1} p_{i,j_1}^p + u_{i,j_2} p_{i,j_2}^p. \tag{23}$$

Recall that $\mathbf{E}[T_i^p \mid U = u]$ takes the form (21) or (22); this, in conjunction with (23) and Lemma 4.2, shows that for all possible u,

$$\mathbf{E}[T_i^p \mid U = u] \leq \gamma(p) \cdot (t_i^{*p} + \mathbf{E}[\mu(X, i) \mid U = u]).$$

Thus,

$$\mathbf{E}[T_i^p] \leq \gamma(p)(t_i^{*p} + \mathbf{E}[\mu(X, i)]) \leq \gamma(p)(t_i^{*p} + \sum_j x_{i,j}^* p_{i,j}^p),$$

where the second inequality follows from Lemma 2.2. So, $\sum_i \mathbf{E}[T_i^p] \leq 2\gamma(p) \cdot T^p$, by (17) and (18). The claims for $\rho(p)$ follow by noting that $\rho(p) \leq (2\gamma(p))^{\frac{1}{p}}$ and substituting $\gamma(p)$ from Lemma 4.1, and by Jensen's inequality which implies that for any non-negative random variable Υ, $\mathbf{E}[\Upsilon] \leq (\mathbf{E}[\Upsilon^p])^{1/p}$. □

5 Multi-criteria optimization for multiple L_p norms and weighted completion time

We now demonstrate that algorithm SchedRound is useful in multi-criteria optimization as well. We present our multi-criteria optimization results for a given collection of integer L_p norms and weighted completion time, in Section 5.2. We then improve upon the results of [5, 9] that pertain to *all* norms $p \geq 1$ for the *restricted assignment* version of the unrelated-parallel-machines problem, in Section 5.3.

The setting of Section 5.2 is as follows. Let S be a set of positive integers and let $T(p)$ be a target value for each $p \in S$. Let W^* be a targeted total weighted completion time. We aim to obtain a schedule such that the L_p norm of the vector of machine-loads is not much more than $T(p)$ for each $p \in S$, and such that the weighted completion time is not much more than W^*. (In some cases, such as in part (1) of the statement of Theorem 5.1, we will not be concerned with the weighted completion time, in which case we set $W^* = \infty$.) Section 5.1 presents a natural convex programming relaxation for this problem; Section 5.2 then develops a deterministic multi-criteria approximation algorithm in which the rounding is basically a derandomization of a modified version of algorithm SchedRound.

5.1 The formulation (MULT)

Given targets $\{T(p) : p \in S\}$ and W^* as in the previous paragraph, the following formulation (MULT) suggests itself. We modify the formulation of Section 4 as follows:

- we retain the constraints (15), (16) and (19);
- we include equations (17) and (18) for each $p \in S$, with the "T" in the right-hand-sides replaced by "$T(p)$";
- we include constraints (8) and (9) from Section 3, with the "z" in the left-hand-sides replaced by "W^*";
- we replace (20) by

$$\forall (i,j) \in \{(i',j') \mid \exists p \in S \text{ such that } p_{i',j'} > T(p)\},\ x_{i,j} = 0.$$

It is easy to see that the resulting formulation (MULT) is indeed a valid integer formulation of the given problem with targets $\{T(p) : p \in S\}$ and W^*. Furthermore, the discussions of Sections 3 and 4 show that the natural continuous relaxation of (MULT) obtained by replacing (19) by "$\forall (i,j), x_{i,j} \geq 0$" is a valid convex formulation of the problem.

5.2 The rounding approach for (MULT)

Given an integer assignment $X = (X_{i,j})$, we set $C_p(X)$ and $W(X)$ to be the L_p norm of the vector of machine-loads and the weighted completion time under X, respectively. Let x^* be a fractional assignment that is feasible for the continuous relaxation of (MULT). Theorem 5.1 essentially uses SchedRound to round x^*. Note that Theorem 3.2 follows as a corollary of claim (4) of Theorem 5.1, by letting the parameter ϵ tend to 0 from above in claim (4) of Theorem 5.1.

Theorem 5.1 *Suppose, for a given problem with target machine-loads $\{T(p) : p \in S\}$ and completion-time target W^*, that x^* is a feasible fractional solution to the continuous relaxation of (MULT). Then, we can derandomize SchedRound in polynomial time to obtain an integer assignment $X = (X_{i,j})$ that achieves any desired one of the following four outcomes:*

1. *For all $p \in S$, $C_p(X) \leq 2.56 \cdot T(p)$;*

2. *$W(X) \leq 3.2 \cdot W^*$ and for all $p \in S$, $C_p(X) \leq 3.2 \cdot T(p)$;*

3. *For any given $\epsilon > 0$, $W(X) \leq \frac{3}{2} \cdot (1+\epsilon) W^*$ and for all $p \in S$, $C_p(X) \leq 2(e + \frac{2}{\epsilon}) \cdot T(p)$, where e is the base of natural logarithms; and*

4. *For any given $\epsilon > 0$ and any given $p \geq \frac{K}{\epsilon^2}$, $W(X) \leq \frac{3}{2}(1+\epsilon)$ and $C_p(X) \leq 2 \cdot T(p)$. (K is some absolute constant here.)*

Proof: We show how to obtain each of the four guarantees claimed in the theorem.

Guarantee 1: We now describe a derandomization of SchedRound, in order to get the guarantee for all $p \in S$. We first recall a few definitions and define new ones. Let $X^{(h)}$ denote the (fractional) assignment vector after iteration h in our derandomized rounding algorithm, with $X^{(0)} \doteq x^*$; let t_i^* denote the fractional load imposed by assignment x^* on machine i. We let x denote an arbitrary assignment vector. For a fixed

machine i, let $\mu_p(x,i) = \sum_j x_{i,j} p_{i,j}^p$. Let X denote the final integral assignment, and let T_i denote the final load on machine i. Let R_i, j_1, and j_2, denote the rounded load and the two floating jobs assigned to i respectively, at the beginning of the first iteration in which i was unprotected. Define $\phi_p(x,i)$ as follows: if $x_{i,j_1} + x_{i,j_2} \in [0,1]$, then

$$\phi_p(x,i) = (1 - x_{i,j_1} - x_{i,j_2}) \cdot R_i^p + x_{i,j_1} \cdot (R_i + p_{i,j_1})^p + x_{i,j_2} \cdot (R_i + p_{i,j_2})^p$$

else if $x_{i,j_1} + x_{i,j_2} \in (1,2]$, then

$$\phi_p(x,i) = (1 - x_{i,j_2}) \cdot (R_i + p_{i,j_1})^p + (1 - x_{i,j_1}) \cdot (R_i + p_{i,j_2})^p + (x_{i,j_1} + x_{i,j_2} - 1) \cdot (R_i + p_{i,j_1} + p_{i,j_2})^p$$

This definition is motivated by the fact that $\phi(X,i) = T_i^p$, just as in (21) and (22). Recall the function $\gamma(\cdot)$ from Lemma 4.1. We next define the quantity $\psi_p(x,i)$ as follows: if $p = 1$, then $\psi_p(x,i) = \phi_p(x,i)$; else if $p > 1$, then $\psi_p(x,i) = \frac{\phi_p(x,i)}{\gamma(p)} - t_i^{*p}$. It follows from Lemma 4.2 that for all $p > 1$ and $\forall i$, $\phi_p(x,i) \leq \gamma(p)(t_i^{*p} + \mu_p(x,i))$, and therefore we have:

$$\psi_p(x,i) \leq \mu_p(x,i). \tag{24}$$

Let $M_1^{(h)}$ and $M_2^{(h)}$ denote the set of protected and unprotected machines respectively immediately after iteration h. Let $Q_p(X^{(h)}) = \sum_{i \in M_1^{(h)}} \mu_p(X^{(h)}, i) + \sum_{i \in M_2} \psi_p(X^{(h)}, i)$. Define $Q(x) = \sum_{p \in S} \frac{Q_p(x)}{f(p) \cdot T(p)^p}$, where the positive values $f(p)$ are chosen such that $\sum_{p \in S} \frac{1}{f(p)} \leq 1$. This implies that $Q(X^{(0)}) \leq 1$.

We are now ready to describe the derandomized version of SchedRound. At iteration h, as in the randomized version, we have two choices of assignment vectors x_1 and x_2 and two scalars $\alpha_1, \alpha_2 \geq 0$ with $\alpha_1 + \alpha_2 = 1$ such that $\alpha_1 x_1 + \alpha_2 x_2 = X^{(h)}$. We choose $X^{(h+1)} \in \{x_1, x_2\}$ such that $Q(X^{(h+1)}) \leq Q(X^{(h)})$. This is always possible because $Q(x)$ is a linear function of the components in x - since we also have $Q(x) = \alpha_1 Q(x_1) + \alpha_2 Q(x_2)$, the minimum of these choices will not increase the value of Q. Next, if a machine i becomes unprotected during some iteration h, then for all $p \in S$, we replace $\mu_p(X^{(h+1)}, i)$ by $\psi_p(X^{(h+1)}, i)$ in the expression for Q. It follows from (24) that this replacement does not increase the value of Q.

Since $Q(X^{(h)})$ is a non-increasing function of h, $Q(X) \leq Q(X^{(0)}) \leq 1$. Hence, it follows that for each $p \in S$, $Q_p(X) \leq f(p)T(p)^p$. We now analyze the final L_p norms for each p. If $p = 1$, the final cost $C_1(X) = \sum_i \phi_1(X,i) = Q_1(X) \leq f(1)T(1)$. We now analyze the values $C_p(X)$ for norms $p > 1$. We first note that all machines become unprotected by the time the algorithm terminates. Hence,

$$\begin{aligned}
(C_p(X))^p &= \sum_i \phi_p(X,i) \\
&= \sum_i \gamma(p)(\psi_p(X,i) + t_i^{*p}) \\
&= \gamma(p)(\sum_i t_i^{*p} + Q_p(X)) \\
&\leq \gamma(p)(T(p)^p + f(p)T(p)^p) \\
&\leq \gamma(p)(f(p)+1)T(p)^p.
\end{aligned}$$

Hence, $C_p(X) \leq (\gamma(p)(1 + f(p)))^{\frac{1}{p}} T(p)$. We are now left to show the choice of values $f(p)$ such that $f(1) = 2.56$, and for all integers $p > 1$, $(\gamma(p)(1 + f(p)))^{\frac{1}{p}} \leq 2.56$ with $\sum_{p=1}^{\infty} \frac{1}{f(p)} \leq 1$. Let $k = 1.28$. We choose $f(p)$ as follows: $f(1) = 2k$; for $p \in \{2,3,4,5,6\}$, $f(p) = \frac{(2k)^p}{\gamma(p)} - 1$; for $p \geq 7$, $f(p) = 4k^p - 1$. By

substituting the minimum achievable value $\gamma(p)$ for each p from Lemma 4.1, we have $(\gamma(p)(1+f(p)))^{\frac{1}{p}} \leq 2.56$ for every integer p. Next, observe that

$$\sum_{p=7}^{\infty} \frac{1}{f(p)} \leq \int_{6}^{\infty} \frac{dr}{4k^r - 1} = \frac{1}{\log k} \cdot \log \frac{4k^6}{4k^6 - 1}.$$

By substituting the value $k = 1.28$, it follows that $\sum_{p=1}^{\infty} \frac{1}{f(p)} \leq 1$.

Guarantees 2, 3, and 4: We now have the problem of simultaneously minimizing the total weighted completion time and L_p norms for the given set of integer-norms S. We proceed quite similarly as in our approach above for Guarantee 1; we will suitably modify our "potential function" $Q(\cdot)$ in order to accommodate the weighted completion-time, and employ Lemmas 2.2 and 3.1 in analyzing the effect of this modification.

Recall the definitions of $Q_p()$ from the proof of Guarantee 1 above. Also recall that the total weighted completion time objective for any integral assignment X is

$$W(X) = \sum_i w_{i,j} X_{i,j} \cdot (p_{i,j} + \sum_{j' \prec_i j} X_{i,j'} p_{i,j'}).$$

We extend this definition to any fractional assignment x:

$$W(x) \doteq \sum_i w_{i,j} x_{i,j} \cdot (p_{i,j} + \sum_{j' \prec_i j} x_{i,j'} p_{i,j'}).$$

We now redefine our combined objective $Q(x)$ as follows:

$$Q(x) = \frac{2W(x)}{3gW^*} + \sum_{p \in S} \frac{Q_p(x)}{f(p) \cdot T(p)^p},$$

where the positive values g and $\{f(p)\}$ satisfy

$$\frac{1}{g} + \sum_{p \in S} \frac{1}{f(p)} \leq 1. \tag{25}$$

We will choose these positive values separately for each of guarantees 2, 3, and 4. Recall the easy fact "$W(x^*) \leq 3W^*/2$" (e.g., from the proof of Theorem 3.2). Since $X^{(0)} = x^*$, x^* is a feasible assignment, and (25) is true, we get that $Q(X^{(0)}) \leq 1$. We now follow the same derandomization strategy as in Guarantee 1: i.e., we choose from two possible choices for $X^{(h+1)}$ such that $Q(X^{(h+1)}) \leq Q(X^{(h)})$. Crucially, we remark that this is possible since, as shown in Lemma 3.1, at every iteration h, the two choices for $X^{(h+1)}$ namely x_1 and x_2, and the scalars $\alpha_1, \alpha_2 \geq 0$ with $\alpha_1 + \alpha_2 = 1$ are such that $\alpha_1 \cdot x_1 + \alpha_2 \cdot x_2 = X^{(h)}$ and $\alpha_1 W(x_1) + \alpha_2 W(x_2) \leq W(X^{(h)})$; this implies that at every iteration of the derandomized algorithm, there exists a choice of $X^{(h+1)}$ such that $Q(X^{(h+1)}) \leq Q(X^{(h)}) \leq 1$. Thus we will have $W(X) \leq \frac{3g}{2} W^*$ for our final integral assignment X. We are now left to show the choice of positive values $\{f(p)\}$ and g such that (25) holds, and such that the tradeoffs claimed in the theorem can be achieved. We show this below for each of guarantees 2, 3, and 4.

Guarantee 2. We fix $k = 1.6$ and let $g = \frac{4k}{3}$. All the values of $f(p)$ remain the same function of k as in the proof guarantee 1: i.e., $f(1) = 2k$, $\forall p \in \{2, \ldots, 6\}$, $f(p) = \frac{(2k)^p}{\gamma(p)} - 1$, and for $p \geq 7$, $f(p) = 4k^p - 1$. It now follows from the arguments for guarantee 1 that (25) holds.

Guarantee 3. Fix $k = e + \frac{2}{\epsilon}$ and $g = 1 + \epsilon$. We let $f(1) = 2k$ and for $p \geq 2$, we let $f(p) = 4k^p - 1$ and $\gamma(p) = 2^{p-2}$ (from Lemma 4.2). This yields an approximation factor of $\frac{3(1+\epsilon)}{2}$ for the completion

time and a factor of $2(e+\frac{2}{\epsilon})$ for each norm $p \in S$. We now have, $\frac{1}{g} + \sum_{p\in S} \frac{1}{f(p)} \leq \frac{1}{1+\epsilon} + \sum_{p=1}^{\infty} \frac{1}{f(p)} \leq 1 - \frac{\epsilon}{2} + \frac{\epsilon}{4} + \frac{1}{\log k} \cdot \log(\frac{4k}{4k-1}) \leq 1 - \frac{\epsilon}{2} + \frac{\epsilon}{4} + \frac{1}{4k-1} \leq 1 - \frac{\epsilon}{2} + \frac{\epsilon}{4} + \frac{\epsilon}{8} \leq 1$.

Guarantee 4. Fix $g = (1+\epsilon)$, and let $f(p) = 1 + \frac{1}{\epsilon}$. We have $\gamma(p) \leq O(\frac{2^p}{\sqrt{p}})$ from Lemma 4.2. Hence for all $p \geq K/\epsilon^2$ with K being a suitably large constant, $\gamma(p) \leq \epsilon 2^{p-2}$. So, $W(X) \leq \frac{3(1+\epsilon)}{2} \cdot W^*$ and $C_p(X) \leq (\gamma(p)(f(p)+1))^{\frac{1}{p}} T(p) \leq 2T(p)$. Furthermore, $\frac{1}{g} + \frac{1}{f(p)} = 1$, which concludes the proof of the theorem. □

5.3 All-norm approximations for the restricted assignment problem

The next theorem pertains to the approximation ratio of SchedRound for the *restricted assignment* problem, where each job j is associated with some number p_j such that for all (i,j), $p_{i,j} \in \{p_j, \infty\}$. That is, each job has a processing time p_j and a subset S_j of the machines; j can only be scheduled on some machine in S_j, and has processing time p_j on all machines in S_j. We will employ Theorem 4.4 to improve on certain results of [5, 9].

As in Azar *et al.* [5], we first obtain the unique fractional solution x^* that is *simultaneously* optimal with respect to all norms $p \geq 1$: this x^* is a strongly-optimal fractional assignment, in the following sense [5]. Consider a fractional assignment x'; let t'_i be the fractional load on machine i induced by x': i.e., $t'_i = \sum_{j\in J} x'_{i,j} p_{i,j}$. Consider any norm $p \geq 1$; the fractional L_p-norm of x' is $L_p(x') = (\sum_{i\in M} (t'_i)^p)^{\frac{1}{p}}$; let $L_p(frac)$ denote the minimum value of the L_p-norm achievable by any valid fractional assignment (i.e., any fractional assignment such that the $x_{i,j}$ are all non-negative, and such that the sum of the $x_{i,j}$ values for each job j is one); let $L_p(int)$ denote the minimum value of L_p-norm achievable by any valid integral assignment. A fractional assignment x^* is strongly-optimal if for any $p \geq 1$, $L_p(x^*) = L_p(frac)$ (i.e, x^* is optimal w.r.t. all the norms). Azar *et al.* [5] show that, given any instance of the restricted assignment problem, there exists a strongly-optimal fractional assignment x^* for the instance; further, such an assignment can be computed in polynomial time. It is also demonstrated in [5] that x^* can be rounded efficiently to get an *absolute* 2-approximation factor w.r.t. every norm $p \geq 1$: this notion of absolute approximation is that each L_p norm is individually at most twice optimal, i.e., at most $2 \cdot L_p(int)$. This result of [5] was also independently shown by Goel and Meyerson [9]. We get an improvement as follows:

Theorem 5.2 *Consider the restricted assignment problem. Given a strongly-optimal fractional assignment x^*, and a fixed norm $p' \in [1,\infty)$, SchedRound can be derandomized in polynomial time to simultaneously yield a $\rho(p') < 2$ absolute approximation w.r.t. norm p' and an absolute 2-approximation w.r.t. all other norms $p \geq 1$, where $\rho(\cdot)$ is the function from Theorem 4.4. That is, the L_p norms C_p of the integral solution constructed, satisfy the following: $C_p \leq 2 \cdot L_p(int)$ for all $p \geq 1$, and $C_{p'} \leq \rho(p') \cdot L_{p'}(int)$.*

Proof: We start by computing a strongly-optimal fractional assignment x^* as in Azar *et al.* [5]. We round this fractional assignment into an integral assignment using algorithm SchedRound. Let C_p be the random variable denoting the L_p norm of the integral assignment produced by our algorithm. We prove below that for all $p \geq 1$, $C_p \leq 2L_p(int)$ *with probability* 1. This claim along with Theorem 4.4 immediately leads to Theorem 5.2 as follows. Consider the fixed norm $p' \in [1,\infty)$; the fractional assignment has an $L_{p'}$ norm value of $L_{p'}(x^*) = L_{p'}(frac) \leq L_{p'}(int)$. Hence, by Theorem 4.4, the integral assignment has an expected value $\mathbf{E}[C_{p'}]$ such that $\mathbf{E}[C_{p'}] \leq \rho(p') L_{p'}(frac) \leq \rho(p') L_{p'}(int)$; here $\rho(p') < 2$ is the approximation ratio in Theorem 4.4. Crucially, since our claim guarantees that $\forall p \geq 1, C_p \leq 2L_p(int)$ with probability 1, we have the conditional expectation $\mathbf{E}[C_{p'} \mid \forall p \geq 1, C_p \leq 2L_p(int)] = \mathbf{E}[C_{p'}] \leq \rho(p') L_{p'}(int)$. Hence, we can derandomize algorithm SchedRound using the method of conditional probabilities (as in Theorem 4.4) to obtain an integral assignment such that $C_{p'} \leq \rho(p') L_{p'}(int)$ and $\forall p \geq 1, C_p \leq 2L_p(int)$.

We now prove that SchedRound produces an integral assignment in which for all $p \geq 1$, $C_p \leq 2L_p(int)$ with probability 1. Let $X = (X_{i,j})$ denote the integral assignment yielded by SchedRound; recall that SchedRound ensures that $X_{i,j}$ can be 1 only if $x^*_{i,j} > 0$. Since we have an instance of restricted assignment and x^* is an optimal fractional assignment w.r.t. all the norms, it follows that $X_{i,j} = 1$ only if $p_{i,j} = p_j$ (and not ∞). We have

$$\begin{aligned}
(C_p)^p &= \sum_i (\sum_{j \in J} X_{i,j} \cdot p_{i,j})^p \\
&< \sum_i \left((\sum_{j \in J} x^*_{i,j} \cdot p_{i,j}) + \max_{j \in J:\ X_{i,j}=1} p_{i,j} \right)^p && (26) \\
&\leq \sum_i 2^{p-1} \cdot \left((\sum_{j \in J} x^*_{i,j} \cdot p_{i,j})^p + (\max_{j \in J:\ X_{i,j}=1} p_j)^p \right) && (27) \\
&\leq 2^{p-1} \cdot \left((\sum_i (\sum_{j \in J} x^*_{i,j} \cdot p_{i,j})^p) + \sum_j p_j^p \right) \\
&\leq 2^{p-1} \cdot (L_p(frac)^p + L_p(int)^p) && (28) \\
&\leq 2^{p-1} \cdot 2L_p(int)^p \\
&= 2^p \cdot L_p(int)^p \\
\text{Hence: } C_p &\leq 2L_p(int)
\end{aligned}$$

In the above derivation, (26) follows from Lemma 2.3(ii). Bound (27) is a consequence of the fact that for all $x \geq 0,\ y \geq 0$ and $p \geq 1$, $(x+y)^p \leq 2^{p-1} \cdot (x^p + y^p)$. Bound (28) follows from the fact that $(\sum_j p_j^p)^{\frac{1}{p}}$ is a lower bound on the L_p norm value of any integral assignment. □

6 Generalizing the Karp *et al.* Procedure, and Applications

We now employ LinAlgRound to develop two additional applications. The 2-approximation for makespan minimization in unrelated parallel machines [18] was extended in [21] as follows. Suppose we are given some numbers $\{c_{i,j}\}$ (where $c_{i,j}$ corresponds, e.g., to the cost of processing job j on machine i), a target makespan T, and a fractional assignment x that satisfies the target makespan (along with the usual important constraints "if $p_{i,j} > T$, then $x_{i,j} = 0$") as well as the constraint $\sum_{i,j} c_{i,j} x_{i,j} = C$ for some C. Then, the algorithm of [21] constructs an integral assignment X such that $\sum_j p_{i,j} X_{i,j} \leq 2T$ for all i, and such that $\sum_{i,j} c_{i,j} X_{i,j} \leq C$. We first describe how Theorem 6.1, a basic rounding theorem of Karp *et al.* [13], can be used to obtain the result of [18]. We then show a probabilistic generalization of this theorem of [13] (Theorem 6.2) which in particular also yields the result of [21]. We also describe an extension (Corollary 6.4) to the setting where we are given multiple cost-objectives and by paying a slightly larger factor for the makespan, we can bound the *absolute* deviation for the additional objectives.

Theorem 6.1 ***([13])*** *Given a matrix $A \in \Re^{m \times n}$, with t denoting $\max_j\{\sum_{i:A_{ij}>0} A_{ij}, -\sum_{i:A_{ij}<0} A_{ij}\}$, and a fractional vector x, we can, in deterministic polynomial time, construct a vector X such that: (i) $\forall j$, $X_j \in \{\lfloor x_j \rfloor, \lceil x_j \rceil\}$, and (ii) $\forall i$, $(AX)_i < (Ax)_i + t$.*

To see how Theorem 6.1 yields the result of [18], consider the LP of [18] for makespan minimization, given a target makespan of T:

(A1) $\forall j$, $\sum_i x_{i,j} \geq 1$;

(A2) $\forall i$, $\sum_j p_{i,j} x_{i,j} \leq T$;

(A3) $0 \leq x_{i,j} \leq 1$; and

(A4) $p_{i,j} > T$ implies $x_{i,j} = 0$.

If we multiply the constraints **(A1)** by $-T$, the parameter t, in the notation of Theorem 6.1, can be taken to be T; therefore there is an integral vector X such that: (i) for each j, $\sum_i -X_{i,j} < 0$, or $\sum_i X_{i,j} \geq 1$ (i.e., job j is assigned to some machine), and (ii) for each i, $\sum_j p_{i,j} X_{i,j} \leq 2T$. We now describe our probabilistic generalization of Theorem 6.1; it is a generalization since it guarantees the additional properties (iii) and (iv):

Theorem 6.2 *Given a matrix $A \in \Re^{m \times n}$, with t denoting $\max_j \{\sum_{i:A_{ij}>0} A_{ij}, -\sum_{i:A_{ij}<0} A_{ij}\}$, and an n-dimensional vector x, we can, in randomized polynomial time, construct a vector X such that: (i) $\forall j$, $X_j \in \{\lfloor x_j \rfloor, \lceil x_j \rceil\}$ with probability 1, (ii) $\forall i$, $(AX)_i < (Ax)_i + t$ with probability 1, and (iii) for each j, $\mathbf{E}[X_j] = x_j$; in particular, given any vector $\vec{c}$, we have $\mathbf{E}[\sum_j c_j X_j] = C \doteq \sum_j c_j x_j$. Furthermore: (iv) a vector X for which (i) and (ii) hold, and for which $\sum_j c_j X_j \leq C$, can be constructed in deterministic polynomial time.*

Proof: We essentially use algorithm LinAlgRand on top of the basic approach of Theorem 6.1 as follows. By subtracting out integer parts, we may assume that $x_j \in [0, 1]$ for all j. We proceed in iterations, each of which further rounds at least one yet-unrounded x_j to 0 or 1. We now describe a typical iteration. Suppose $\mathcal{L}$ is the current linear system (when we start, $\mathcal{L}$ is the original system "$Ax = b$"). Recall the `Simplify` operation related to LinAlgRand. We first run `Simplify` to make $\mathcal{L}$ canonical. Now, it is shown in [13] that: (a) either $\mathcal{L}$ is under-determined, or (b) there exists a row i (which can be found efficiently) in our system of constraints $\mathcal{L}$ such that no matter how we round the yet-unrounded variables to get a final (rounded) vector X, we will have $(AX)_i < b_i + t$. If (b) holds, we keep discarding such rows i from consideration (since they are "safe" from now on), until case (a) holds. And now that the system is under-determined, we apply `RandStep`. This completes the description of an iteration.

Lemma 2.4 easily shows properties (i), (ii) and (iii) of the theorem. Property (iv) also follows directly by derandomizing this algorithm – i.e., making each application of `RandStep` deterministic – using the method of conditional probabilities as follows. Suppose S is the current set of yet-unrounded variables, and that `RandStep` has the choice of moving along directions $\vec{r}$ or $-\vec{r}$: then, we choose $\vec{r}$ if $\sum_{j \in S} c_j r_j \leq 0$, and choose $-\vec{r}$ otherwise. □

Recall the argument above (using the constraints **(A1)**–**(A4)**), about how Theorem 6.1 yields the result of [18]; Theorem 6.2 can be similarly seen to imply the result of [21].

We now extend Theorem 6.2 as follows. Suppose, in addition to the linear system $Ax = b$, we have $\ell + 1$ additional linear constraints $c^{(k)} \cdot x = d_k$, $k = 0, 1, \ldots \ell$. We prove that these additional constraints can be approximated well, by incurring a slightly larger error in the system "$Ax = b$":

Corollary 6.3 *Suppose we are given: (i) a matrix $A \in \Re^{m \times n}$, with the parameter t as in Theorem 6.2; (ii) an n-dimensional vector x, (iii) $\ell + 1$ linear constraints $c^{(k)} \cdot x = d_k$, $k = 0, 1, \ldots \ell$, that are satisfied by the given x, and (iv) a positive real ϵ. We can, in deterministic polynomial time, construct a vector X such that:*

(a) $\forall j$, $X_j \in \{\lfloor x_j \rfloor, \lceil x_j \rceil\}$;

(b) $\forall i,\ (AX)_i < (Ax)_i + t(1+\epsilon)$;

(c) *for all* $k = 1, 2, \ldots, \ell$, $|c^{(k)} \cdot X - d_k| \leq (1+\epsilon)M_k\ell/\epsilon$, *where* M_k *is the maximum absolute value of any coefficient in* $c^{(k)}$ *(note that the case* $k = 0$ *is excluded here, and is handled next in part (d)); and*

(d) $c^{(0)} \cdot X \leq d_0$.

Proof: We will first rescale the given system with constraints "$Ax = b$" and "$c^{(k)} \cdot x = d_k,\ k = 1, \ldots \ell$" (note that the constraint for $k = 0$ is excluded here) as follows: scale the system "$Ax = b$" by $1/((1+\epsilon)t)$, and replace the constraints "$c^{(k)} \cdot x = d_k,\ k = 1, \ldots \ell$" by the following 2ℓ constraints:

$$\begin{aligned} \frac{\epsilon}{(1+\epsilon)\cdot\ell\cdot M_k} \cdot (c^{(k)} \cdot x) &\leq \frac{\epsilon}{(1+\epsilon)\cdot\ell\cdot M_k} \cdot d_k;\ k = 1, \ldots \ell; \\ -\frac{\epsilon}{(1+\epsilon)\cdot\ell\cdot M_k} \cdot (c^{(k)} \cdot x) &\leq -\frac{\epsilon}{(1+\epsilon)\cdot\ell\cdot M_k} \cdot d_k;\ k = 1, \ldots \ell. \end{aligned}$$

Note that the parameter t for this system, in the notation of Theorem 6.2, is at most

$$\frac{t}{(1+\epsilon)t} + \sum_{k=1}^{\ell} \frac{\epsilon M_k}{(1+\epsilon)\cdot\ell\cdot M_k} = 1,$$

where the "M_k" in the numerator of the second term in the l.h.s. comes from the fact that M_k is the maximum absolute value of any coefficient in $c^{(k)}$. Now apply part (iv) of Theorem 6.2, with the constraint "$c^{(0)} \cdot x = d_0$" playing the role of the constraint "$\sum_j c_j x_j = C$" of Theorem 6.2. We now verify that the resultant vector X satisfies the guarantees of this corollary. Guarantees (a) and (d) directly follow from Theorem 6.2. Guarantee (b) follows from Theorem 6.2, via a rescaling by $(1+\epsilon)t$. Similarly, Guarantee (c) follows from rescaling by the quantities $\frac{(1+\epsilon)\cdot\ell\cdot M_k}{\epsilon}$. □

Using the constraints **(A1)**–**(A4)** once again, Corollary 6.3 specializes to the following result on multi-objective scheduling:

Corollary 6.4 *Suppose we are given an instance of unrelated-machine-scheduling the cost of processing job* j *on machine* i*), a target makespan* T*, and a fractional assignment* x *that satisfies the target makespan, as well as the constraints "if* $p_{i,j} > T$*, then* $x_{i,j} = 0$*". Suppose further that* x *satisfies the* $\ell + 1$ *linear constraints* $c^{(k)} \cdot x = d_k,\ k = 0, 1, \ldots \ell$*. Then, for any given* $\epsilon > 0$*, we can construct an integral schedule* $X = (X_{i,j})$ *such that:*

(a) *the makespan is at most* $(2+\epsilon)T$;

(b) *for all* $k = 1, 2, \ldots, \ell$, $|c^{(k)} \cdot X - d_k| \leq (1+\epsilon)M_k\ell/\epsilon$, *where* M_k *is the maximum absolute value of any coefficient in* $c^{(k)}$*; and*

(c) $c^{(0)} \cdot X \leq d_0$.

Note in particular that for bounded M_k, ℓ and ϵ, we get a constant *additive error* for the additional constraints; we are not aware of any other method that can yield this, even for small constants ℓ.

Finally, we consider the problem of unrelated parallel machine scheduling with resource dependent processing times. This is a variant of the standard unrelated parallel machine scheduling, where the processing times $p_{i,j}$ of any machine-job pair can be reduced by utilizing a renewable resource (such as additional workers) that can be distributed over the jobs. Specifically, a maximum number of k units of a resource may be used to speed up the jobs, and the available amount of k units of that resource must

not be exceeded at any time. Grigoriev *et al.* [11] presented a $4 + 2\sqrt{2}$ approximation for this problem. A direct application of Theorem 6.2 yields an assignment of jobs and resources to machines; combined with the scheduling algorithm of [11], we developed in the conference version of this work [15] a 4-approximation for the problem. Our work has been further built upon in [12], leading to a 3.75-approximation. We refer the reader to [12] for complete details, but the basic idea for our improvement exactly follows from the fact that Theorem 6.2 is able to incorporate one additional linear constraint (i.e., via $\vec{c}$), without losing any of the guarantees of Theorem 6.1.

7 Convergence to fairness

Quite a number of our bounds are of the following type. Some random variable X:

(P1) has "low" expectation μ (often equal to the corresponding LP-value), and

(P2) is at most some value a with probability 1.

Two examples of this are as follows: (a) as seen from Sections 2 and 3, SchedRound ensures that the final load on machine i, $\sum_j p_{i,j} X_{i,j}$, equals its LP-value t_i^* in expectation, and is at most $2t_i^*$ with probability one; (b) properties (ii) and (iii) of Theorem 6.2 show that for any row i, $(AX)_i$ equals $(Ax)_i$ in expectation, and is less than $(Ax)_i + t$ with probability one. We often use only (P2) in bounding our approximation guarantees; in this short section, we observe that combined usage of (P1) and (P2) easily leads to fairness guarantees under *multiple* executions.

Consider, for instance, the makespan-minimization setting described in application (iv) in the introduction; the reader is asked to recall the notation used therein. From the discussion of the previous paragraph, we have that for any i and k, $Z_{i,k}$ lies in $[0, 2]$, and has mean at most 1. Thus, since $Z_{i,k}$ is a bounded random variable, a standard application of the Chernoff bounds shows that for any particular i,

$$\Pr[\overline{Z}_i(N) > (1 + \epsilon)] \ll 1/m,$$

if $N \geq K(\log m)/\epsilon^2$. Thus, a union bound over all m indices i gives simultaneous fairness for all machines, with high probability. Similar fairness considerations apply to any random variable of the type considered in the previous paragraph.

8 Conclusions

We have presented a new approach to scheduling through SchedRound, which is a rounding algorithm based on linear algebra and randomization. SchedRound offers a unified way to tackle a number of different objectives in scheduling jobs on unrelated parallel machines. One natural question left open is to improve the specific bounds developed in this paper: e.g., can we do better for at least one of the pair (makespan, weighted completion time) of objectives? Also, could linear-algebraic considerations help with rounding for semidefinite programs, where variants of the seminal random-hyperplane technique [10] appear to be the foremost tools of choice?

As mentioned in Section 6, our work has been used to develop improved approximation algorithms for scheduling problems where processing times are a function of the number of resources deployed [12]. Our methods have also been put to use in [16] for game-theoretic issues in scheduling. We anticipate further such applications in the field of approximation algorithms.

Acknowledgments. We thank David Shmoys for valuable discussions, Cliff Stein for introducing us to [24], and Yossi Azar for sending us an early version of [4]. We are thankful to the FOCS 2005 and JACM referees for their valuable comments. V. S. A. Kumar and M. V. Marathe thank their external collaborators and members of the Network Dynamics and Simulation Science Laboratory (NDSSL) for their suggestions and comments. This research of Kumar and Marathe has been partially supported by NSF NeTS Grant CNS-0626964, NSF HSD Grant SES-0729441, CDC Center of Excellence in Public Health Informatics Grant 2506055-01, NIH-NIGMS MIDAS Project 5 U01 GM070694-05, DTRA CNIMS Grant HDTRA1-07-C-0113 and NSF NeTS Grant CNS-0831633. S. Parthasarathy's research has been supported in part by NSF Award CCR-0208005 and NSF ITR Award CNS-0426683. A. Srinivasan's research has been supported in part by NSF Award CCR-0208005, NSF ITR Award CNS-0426683, and NSF Award CNS-0626636.

References

[1] Alon, N., Azar, Y., Woeginger, G. J., and Yadid, T. Approximation schemes for scheduling. In *Proc. ACM-SIAM Symposium on Discrete Algorithms* (1997), pp. 493–500.

[2] Aslam, J., Rasala, A., Stein, C., and Young, N. Improved bicriteria existence theorems for scheduling. In *Proc. ACM-SIAM Symposium on Discrete Algorithms* (1999), pp. 846–847.

[3] Awerbuch, B., Azar, Y., Grove, E. F., Kao, M.-Y., Krishnan, P., and Vitter, J. S. Load balancing in the L_p norm. In *Proc. IEEE Symposium on Foundations of Computer Science* (1995), pp. 383–391.

[4] Azar, Y., and Epstein, A. Convex programming for scheduling unrelated parallel machines. In *Proc. of the ACM Symposium on Theory of Computing* (2005), pp. 331–337.

[5] Azar, Y., Epstein, L., Richter, Y., and Woeginger, G. J. All-norm approximation algorithms. *J. Algorithms 52*, 2 (2004), 120–133.

[6] Azar, Y., and Taub, S. All-norm approximation for scheduling on identical machines. In *Proc. Scandinavian Workshop on Algorithm Theory* (2004), pp. 298–310.

[7] Chandra, A. K., and Wong, C. K. Worst-case analysis of a placement algorithm related to storage allocation. *SIAM J. on Computing 4*, 3 (1975), 249–263.

[8] Gandhi, R., Khuller, S., Parthasarathy, S., and Srinivasan, A. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM 53* (2006), 324–360.

[9] Goel, A., and Meyerson, A. Simultaneous optimization via approximate majorization for concave profits or convex costs. Tech. Report CMU-CS-02-203, December 2002, Carnegie-Mellon University.

[10] Goemans, M. X., and Williamson, D. P. Improved approximation algorithms for Maximum Cut and Satisfiability problems using Semidefinite Programming. *Journal of the ACM 42* (1995), 1115–1145.

[11] Grigoriev, A., Sviridenko, M., and Uetz, M. Unrelated parallel machine scheduling with resource dependent processing times. In *Proc. Integer Programming and Combinatorial Optimization (IPCO)* (2005), pp. 182–195.

[12] Grigoriev, A., Sviridenko, M., and Uetz, M. Machine scheduling with resource dependent processing times. *Mathematical Programming 110* (2007), 209–228.

[13] Karp, R. M., Leighton, F. T., Rivest, R. L., Thompson, C. D., Vazirani, U. V., and Vazirani, V. V. Global wire routing in two-dimensional arrays. *Algorithmica* (1987), 113–129.

[14] Kleinberg, J., Tardos, E., and Rabani, Y. Fairness in routing and load balancing. *J. Comput. Syst. Sci. 63*, 1 (2001), 2–20.

[15] Kumar, V. S. A., Marathe, M. V., Parthasarathy, S., and Srinivasan, A. Approximation algorithms for scheduling on multiple machines. In *Proc. IEEE Symposium on Foundations of Computer Science* (2005), pp. 254–263.

[16] Lavi, R., and Swamy, C. Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. In *Proc. ACM Conference on Electronic Commerce* (2007), pp. 252–261.

[17] Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R., and Shmoys, D. B. *Sequencing and scheduling: algorithms and complexity.* Elsevier, 1993.

[18] Lenstra, J. K., Shmoys, D. B., and Tardos, E. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming* (1990), 259–271.

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

[20] Schuurman, P., and Woeginger, G. J. Polynomial time approximation algorithms for machine scheduling: Ten open problems. *J. Scheduling* (1999), 203–213.

[21] Shmoys, D. B., and Tardos, E. An approximation algorithm for the generalized assignment problem. *Mathematical Programming* (1993), 461–474.

[22] Skutella, M. Convex quadratic and semidefinite relaxations in scheduling. *Journal of the ACM 46*, 2 (2001), 206–242.

[23] Smith, W. E. Various optimizers for single-stage production. *Nav. Res. Log. Q.* (1956), 59–66.

[24] Stein, C., and Wein, J. On the existence of schedules that are near-optimal for both makespan and total weighted completion time. *Operations Research Letters 21* (1997), 115–122.

Appendix

A Proofs and Auxiliary Results

Proof: **(For Lemma 4.1)** Consider any $p \geq 1$. Let $N'(a,\lambda) = pa \cdot (1+\lambda)^{p-1}$ and $D'(a,\lambda) = pa \cdot (1+a\lambda)^{p-1} + pa\lambda^{p-1}$ be the derivatives of $N(a,\lambda)$ and $D(a,\lambda)$ respectively w.r.t. λ. Observe that $N(a,\lambda) = N(a,0) + \int_0^\lambda N'(a,\lambda)d\lambda$ and $DN(a,\lambda) = D(a,0) + \int_0^\lambda D'(a,\lambda)d\lambda$. Hence,

$$\max_{a,\lambda} \frac{N(a,\lambda)}{D(a,\lambda)} \leq \max\left\{\frac{N(a,0)}{D(a,0)}, \max_{a,\lambda} \frac{N'(a,\lambda)}{D'(a,\lambda)}\right\}.$$

We have $\max_{a,\lambda} \frac{N'(a,\lambda)}{D'(a,\lambda)} \leq \frac{(1+\lambda)^{p-1}}{1+\lambda^{p-1}}$ which is maximized when $\lambda = 1$. Hence $\max_{a,\lambda} \frac{N(a,\lambda)}{D(a,\lambda)} \leq \max\{1, 2^{p-2}\}$. Thus the lemma holds for the first two cases.

Now suppose p is sufficiently large. Observe that

$$\frac{N(a,\lambda)}{D(a,\lambda)} \leq \Lambda \doteq \frac{a(1+\lambda)^p + 1 - a}{1 + pa\lambda + a\lambda^p}.$$

We now analyze Λ. Both the numerator and denominator of Λ are linear functions of a, and the denominator is positive; so, Λ is maximized when $a \in \{0,1\}$. When $a = 0$, $\Lambda = 1$ and the lemma holds. Hence, it is enough to show that $F(\lambda) \doteq \frac{(1+\lambda)^p}{1+p\lambda+\lambda^p}$ is at most $O(2^p/\sqrt{p})$. If $\lambda \leq \frac{1}{2}$, then $F(\lambda) \leq \left(\frac{3}{2}\right)^p$; else if $\lambda \in [\frac{1}{2}, 1]$, the denominator of $F(\lambda)$ is at least $\frac{p}{2}$ and the numerator is at most 2^p. Hence the lemma holds if $\lambda \in [0,1]$. Assume next that $\lambda > 1$, and set $\lambda = \frac{1+\epsilon}{1-\epsilon}$ for some positive ϵ. Then,

$$F(\lambda) \leq \frac{2^p}{p(1-\epsilon)^p + (1+\epsilon)^p}. \tag{29}$$

The denominator here is minimized when

$$\frac{1+\epsilon}{1-\epsilon} = p^{\frac{1}{p-1}} = e^{(\ln p)/(p-1)} = 1 + \frac{\ln p}{p} \pm O((\frac{\ln p}{p})^2).$$

We thus take $\epsilon = \frac{\ln p}{2p} \pm O((\frac{\ln p}{p})^2)$. This implies that $1+\epsilon = 1+\frac{\ln p}{2p} \pm O((\frac{\ln p}{p})^2)$ and $1-\epsilon = 1-\frac{\ln p}{2p} \pm O((\frac{\ln p}{p})^2)$. Substituting back these values in (29) yields

$$\begin{aligned} F(\lambda) &\leq \frac{2^p}{p \cdot [1 - (\ln p)/(2p) \pm O(((\ln p)/p)^2)]^p + [1 + (\ln p)/(2p) \pm O(((\ln p)/p)^2)]^p} \\ &= \Theta(2^p/(p \cdot 1/\sqrt{p} + \sqrt{p})) \\ &= \Theta(2^p/\sqrt{p}). \end{aligned}$$

Thus we have the lemma's claim for large p.

Finally, suppose $p \in S = \{2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6\}$. We use numerical techniques to obtain tighter bounds on $\gamma(p)$. Define $f(a,\lambda) \doteq (1+a\lambda)^p$ and $g(a,\lambda) \doteq \frac{1}{\gamma(p)} + a \cdot \frac{(1+\lambda)^p - 1 - \gamma(p)\lambda^p}{\gamma(p)}$. For each $p \in S$, it suffices to show for all $(a,\lambda) \in [0,1] \times [0,\infty)$ that

$$f(a,\lambda) \geq g(a,\lambda). \tag{30}$$

For any fixed λ, $f(a,\lambda)$ is a convex function of a while $g(a,\lambda)$ is linear. Assume $\gamma(p) > 1$. Hence, $f(0,\lambda) \geq g(0,\lambda)$ and $f(1,\lambda) \geq g(1,\lambda)$. Hence, if (30) is violated, then the straight line $g(a,\lambda)$ intersects the convex function $f(a,\lambda)$ at two distinct values of $a \in (0,1)$. In this case, by Lagrange's Mean Value Theorem, there exists an $a \in (0,1)$ such that $f'(a,\lambda)$ (derivative w.r.t. a), i.e., $p\lambda(1+a\lambda)^{p-1}$, equals $\frac{(1+\lambda)^p - 1 - \gamma(p)\lambda^p}{\gamma(p)}$. Let $a^*(\lambda)$ be this value which can be obtained by solving this equation for a. Since f is strictly convex as a function of a, $f(a^*(\lambda),\lambda) < g(a^*(\lambda),\lambda)$. The above arguments yield us the following strategy for choosing $\gamma(p)$. We choose $\gamma(p)$ such that one of the following conditions hold for $\lambda \in [0, 15]$.

1. $a^*(\lambda) < 0$. In this case both the intersection points of $g(a,\lambda)$ and $f(a,\lambda)$ are at values $a \leq 0$. Hence, $g(a,\lambda) \leq f(a,\lambda)$ for all values of $a \in [0,1]$ and our claims hold.

2. $f(a^*(\lambda),\lambda) - g(a^*(\lambda),\lambda) \geq 0$. In this case the functions do not intersect within the ranges of a and λ that are of interest to us.

For the choices of $\gamma(p)$ in this lemma, one of these two conditions occurs and this can be verified numerically by plotting the above functions of λ in the range $\lambda \in [0, 15]$. We restrict ourselves to $\lambda \in [0, 15]$ since the fraction $\frac{N(a,\lambda)}{D(a,\lambda)}$ for the values of $\lambda > 15$ can be easily seen to be within $\gamma(p)$ for the values of p considered here. This completes the proof of the lemma. □

Proof: **(For Lemma 4.2)** Recall that p, λ_1, λ_2 are arbitrary but fixed positive values with $p > 1$; also, λ_0 is some non-negative constant. In all the cases below, we assume w.l.o.g. that $\lambda_1 \geq \lambda_2$. Let us first

dispose of the easy case where $\lambda_0 = \lambda_2 = 0$. In this case, $N = a_1\lambda_1^p$ regardless of the value of $a_1 + a_2$, and $D = (a_1^p + a_1) \cdot \lambda^p$; so, $N \leq D$. Note from Lemma 4.1 that $\gamma(p) \geq 1$, since we can always set $a = \lambda = 0$ in Lemma 4.1. So, $N \leq \gamma(p) \cdot D$ if $\lambda_0 = \lambda_2 = 0$.

Hence, we assume from now on that $\lambda_0 + \lambda_2 > 0$. We analyze three possible cases next.

Case 1: $a_1 + a_2 = 1$**.** Since $a_1 + a_2 = 1$, $\lambda_1 \geq \lambda_2$ and $\lambda_0 + \lambda_2 > 0$, D is nonzero. We have

$$\begin{aligned}\frac{N}{D} &= \frac{a_1(\lambda_0+\lambda_1)^p + (1-a_1)(\lambda_0+\lambda_2)^p}{(\lambda_0+\lambda_2+a_1(\lambda_1-\lambda_2))^p + a_1\lambda_1^p + (1-a_1)\lambda_2^p} \\ &\leq \frac{a_1(\lambda_0+\lambda_2+(\lambda_1-\lambda_2))^p + (1-a_1)(\lambda_0+\lambda_2)^p}{(\lambda_0+\lambda_2+a_1(\lambda_1-\lambda_2))^p + a_1(\lambda_1-\lambda_2)^p}\end{aligned}$$

By scaling both the numerator and denominator of the latter fraction by $(\lambda_0 + \lambda_2)^p > 0$, and by letting $\lambda = (\lambda_1 - \lambda_2)/(\lambda_0 + \lambda_2)$ (which is non-negative since $\lambda_1 \geq \lambda_2$ by assumption), the fraction is seen to assume the form as in Lemma 4.1. Hence the lemma holds.

Case 2: $a_1 + a_2 \geq 1$**.** In this case, D is nonzero and

$$\frac{N}{D} = \frac{(a_1+a_2-1)(\lambda_0+\lambda_2+\lambda_1)^p + (1-a_1)(\lambda_0+\lambda_2)^p + (1-a_2)(\lambda_0+\lambda_1)^p}{(\lambda_0+a_1\lambda_1+a_2\lambda_2)^p + a_1\lambda_1^p + a_2\lambda_2^p}. \tag{31}$$

Let $\lambda_0 + a_1\lambda_1 + a_2\lambda_2 = \phi$, and hold ϕ fixed. Thus, we view a_1 and a_2 as variables subject to the following four linear constraints: (i) $0 \leq a_1 \leq 1$, (ii) $0 \leq a_2 \leq 1$, (iii) $a_1 + a_2 \geq 1$, and (iv) $\lambda_0 + a_1\lambda_1 + a_2\lambda_2 = \phi$, where ϕ is fixed. Now, the fraction in (31) becomes a rational function of a_1 and a_2 with a positive denominator; so, it is maximized when (at least) two of the constraints (i)-(iv) are met with equality. If $a_1 + a_2 = 1$, then Case 1 occurs and the lemma holds. Otherwise, either a_1 or a_2 is equal to 1, and $a_1 + a_2 > 1$. Suppose $a_2 = 1$ and $a_1 > 0$. We have

$$\begin{aligned}\frac{N}{D} &= \frac{a_1(\lambda_0+\lambda_2+\lambda_1)^p + (1-a_1)(\lambda_0+\lambda_2)^p}{(\lambda_0+\lambda_2+a_1\lambda_1)^p + a_1\lambda_1^p + \lambda_2^p} \\ &\leq \frac{a_1(\lambda_0+\lambda_2+\lambda_1)^p + (1-a_1)(\lambda_0+\lambda_2)^p}{(\lambda_0+\lambda_2+a_1\lambda_1)^p + a_1\lambda_1^p}.\end{aligned}$$

If we scale both the numerator and denominator of the latter fraction by $(\lambda_0 + \lambda_2)^p > 0$, it assumes the same form as in Lemma 4.1. Hence the lemma holds. An identical argument applies when $a_1 = 1$.

Case 3: $a_1 + a_2 \leq 1$**.** If $\lambda_0 = 0$, then $N \leq D$ and again, we get $N \leq \gamma(p) \cdot D$. So, we assume that $\lambda_0 > 0$, which implies that $D > 0$. We proceed as in Case 2, holding ϕ fixed; the only change is that constraint (iii) of Case 2 now becomes "$a_1 + a_2 \leq 1$". Once again, N/D is maximized when at least two of the four constraints (i)-(iv) are met with equality, and we are reduced to Case 1 if $a_1 + a_2 = 1$. So, the only remaining case is where one among a_1 and a_2 is zero, and the other is strictly smaller than 1. So, suppose $a_2 = 0$ and $0 \leq a_1 < 1$. We get

$$\frac{N}{D} = \frac{a_1(\lambda_0+\lambda_1)^p + (1-a_1)\lambda_0^p}{(\lambda_0+a_1\lambda_1)^p + a_1\lambda_1^p},$$

which assumes the same form as in Lemma 4.1 upon scaling the numerator and denominator by λ_0^p. An identical argument applies when $a_1 = 0$. □