

# Approximation Algorithms for Scheduling on Multiple Machines

V.S. Anil Kumar<sup>1</sup>

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

Srinivasan Parthasarathy<sup>2</sup>

Department of Computer Science  
University of Maryland  
College Park, MD 20742  
Email: sri@cs.umd.edu

Madhav V. Marathe

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

Aravind Srinivasan<sup>2</sup>

Dept. of Computer Science & UMIACS  
University of Maryland  
College Park, MD 20742  
Email: srin@cs.umd.edu

## 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. We obtain 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 ( $3/2$  and  $2$ , respectively); (ii) better-than-two approximation guarantees for scheduling under the  $L_p$  norm for all  $1 < p < \infty$ , improving on the 2-approximation algorithms of Azar & Epstein; and (iii) the first constant-factor multicriteria approximation algorithms that handle the weighted completion-time and any given collection of integer  $L_p$  norms. Our algorithm yields a common generalization of rounding theorems due to Karp et al. and Shmoys & Tardos; among other applications, this yields an improved approximation for scheduling with resource-dependent processing times studied by Grigoriev et al.

## 1 Introduction

The complexity and approximability of scheduling problems for multiple machines is an area of active research

[14, 17]. 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 considered 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  $L_\infty$  norm of the machine-loads) [15, 18, 19, 5]. There is no 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  $i$ th objective, for each  $i$ . (One typical goal here is to minimize  $\max_i \lambda_i$ .) Most of the best results 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 [15, 18], and standard randomized rounding [16] as applied to specific problems in [19, 5].

In this work, 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

<sup>1</sup>Work done while at the Los Alamos National Laboratory.

<sup>2</sup>Research supported in part by NSF Award CCR-0208005 and NSF ITR Award CNS-0426683.

makespan objectives, our approach yields a bicriteria approximation with the best known guarantees for both these objectives. We start by presenting our applications, and then discuss our rounding technique.

(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  $3/2$  [19] and  $2$  [15], respectively. We construct schedules that achieve these bounds *simultaneously*: if there exists a schedule with (weighted completion-time, makespan)  $\leq (C, T)$  coordinatewise, 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 and Wein [20] and Aslam *et al.* [3]: e.g., we would get  $\leq (2.7C, 3.6T)$  using the methods of [20]. More importantly, note that if we can improve one component of our pair  $(1.5, 2)$  (while worsening the other arbitrarily), we would improve on 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  $L_\infty$  norm of the machine loads, and that the  $L_1$  norm is easily minimizable. The  $L_p$  norm of the machine loads, for  $1 < p < \infty$ , interpolates between these “minmax” and “minsum” criteria. (See, e.g., [6] for an example that motivates the  $L_2$  norm in this context.) A very recent breakthrough of [5] improves upon the  $\Theta(p)$ -approximation for minimizing the  $L_p$  norm of machine loads [4], by presenting a 2-approximation for each  $p > 1$ , and a  $\sqrt{2}$ -approximation for  $p = 2$ . Our algorithm further improves upon [5] 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, 1.436, 1.46, and 1.485 for  $p = 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5$  and  $6$  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 [8, 1, 6, 7, 10, 13] in various special cases of unrelated parallel machines. For general 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  $L_\infty$  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  $p_1, p_2, \dots, p_r$ , suppose we are given that there exists a schedule in which: (a) for each  $i$ , the  $L_{p_i}$  norm of the ma-

chine 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 [6, 10].

(iv). **Generalization of the Karp *et al.* Rounding Theorem.** A basic result of Karp *et al.* [12], shows that if  $A \in \Re^{m \times n}$  is a “column-sparse” matrix, then for any given real vector  $x = (x_1, x_2, \dots, x_n)^T$ , we can efficiently find a *rounded* counterpart  $y = (y_1, y_2, \dots, y_n)^T$  such that  $\|Ay - Ax\|_\infty$  is “small”. Our approach leads to a probabilistic generalization of this theorem, that achieves the same bound on  $\|Ay - Ax\|_\infty$  with probability 1, with the additional property that for all  $j$ ,  $\mathbf{E}[y_j] = x_j$ . This yields the result of [18]; furthermore, we use the ideas behind this generalization to obtain new multicriteria approximations in the setting of [12]. We also show a direct application of our probabilistic generalization to obtain an improved 4-approximation algorithm for the problem of unrelated parallel machine scheduling with resource-dependent processing times. The current-best approximation known for this problem is  $4 + 2\sqrt{2}$  due to Grigoriev *et al.* [11]. As another application, we show that our main rounding algorithm can be used to obtain a 2-approximate, randomized strategyproof mechanism for the  $Q||C_{max}$  problem, matching the bound of Archer [2].

**Our approach in brief.** 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  $j$ . 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  $X_{i,j}^{(h)}$  denote the random value of  $x_{i,j}$  at the end of iteration  $h$ . For one, we maintain the invariant that  $\mathbf{E}[X_{i,j}^{(h)}] = x_{i,j}^*$  for all  $i, j$  and  $h$ . Second, we “protect” each machine  $i$  almost until the end: the load  $\sum_j p_{i,j} X_{i,j}^{(h)}$  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 [16] and those of [15, 18]. Importantly, while our algorithm is fundamentally based on linear systems, in Lemma 4, we show 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.

Thus, our main algorithm 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 [15, 18] in [5] for general  $L_p$  norms, and independent randomized rounding [16] for weighted completion time in [19] and for the special case of the  $L_2$  norm in [5]. Our algorithm unifies and strengthens all these results. Furthermore, it can be made to work simultaneously with differing objective functions such as weighted completion-time and  $L_p$  norms of machine loads, thus leading to simultaneous multicriteria guarantees. We thus expect our approach to be of use in further contexts as well. Our main algorithm is presented in Section 2, followed by the applications. Due to the lack of space, some of our proofs are deferred to the full version of this paper.

## 2 The Main Rounding Algorithm

We now present our rounding algorithm which 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. 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  $x = x^*$ . Our rounding 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; throughout, we will *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.

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

Our algorithm 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 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, i.e., if there exist machines  $i_1, i_2$  such that  $x_{i_1,j}, x_{i_2,j} \in (0, 1)$ . 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, initially we might 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 above, and recall that  $v > m' + n'$ . Consider the following linear system: **(E1)**  $\forall j \in J', \sum_{i \in M} x_{i,j} = 1$ ; and **(E2)**  $\forall i \in M', \sum_{j \in J'} x_{i,j} \cdot p_{i,j} = \sum_{j \in J'} X_{i,j}^{(h)} \cdot p_{i,j}$ . Note that we only have constraints **(E2)** corresponding to non-singleton machines. 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 **(E1), (E2)** exceeds the number of constraints  $n' + m'$ ; so, there exists a  $v$ -dimensional unit vector  $r$  which can be computed in polynomial time such that starting at point  $P$  and moving along  $r$  or  $-r$  does not violate **(E1)** or **(E2)**. Let  $\alpha$  and  $\beta$  be the strictly-positive quantities such that starting at point  $P$ ,  $\alpha$  and  $\beta$  are the minimum distances to be traveled along directions  $r$  and  $-r$  respectively before one of the variables gets rounded to 0 or 1. We now obtain  $X^{(h+1)}$  as follows. As mentioned before, all values  $X^{(h)}$  which lie in  $\{0, 1\}$ , remain unchanged. For the remaining coordinates, i.e., for the projection  $X_V^{(h+1)}$  of  $X^{(h+1)}$  along the coordinates  $V$ , we do the following: with probability  $\frac{\beta}{\alpha+\beta}$ , set  $X_V^{(h+1)} = X_V^{(h)} + \alpha \cdot r$ ; with the complementary probability of  $\frac{\alpha}{\alpha+\beta}$ , set  $X_V^{(h+1)} = X_V^{(h)} - \beta \cdot r$ .

This way, it is easy to observe that the new system  $X^{(h+1)}$  still satisfies **(E1)** and **(E2)**, has rounded at least one further variable, and also satisfies  $\mathbf{E}[X_{i,j}^{(h+1)}] = X_{i,j}^{(h)}$  (for all  $i, j$ ).

**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.* [9]. 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  $\alpha$  and  $\beta$  as follows.

$$\begin{aligned} \alpha &= \min\{\gamma > 0 : ((\exists(i, j) \in \mathcal{M}_1 : X_{i,j}^{(h)} + \gamma = 1) \\ &\quad \vee (\exists(i, j) \in \mathcal{M}_2 : X_{i,j}^{(h)} - \gamma = 0))\}; \\ \beta &= \min\{\gamma > 0 : ((\exists(i, j) \in \mathcal{M}_1 : X_{i,j}^{(h)} - \gamma = 0) \\ &\quad \vee (\exists(i, j) \in \mathcal{M}_2 : X_{i,j}^{(h)} + \gamma = 1))\}. \end{aligned}$$

We execute the following randomized step, which rounds at least one variable to 0 or 1:

$$\begin{aligned} &\text{With probability } \beta/(\alpha + \beta), \text{ set } X_{i,j}^{(h+1)} := \\ &X_{i,j}^{(h)} + \alpha \text{ for all } (i, j) \in \mathcal{M}_1, \text{ and } X_{i,j}^{(h+1)} := \end{aligned}$$

$X_{i,j}^{(h)} - \alpha$  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 Phase 2, and of our 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 (E2), we will have, for any given value of  $X^{(h)}$ , that

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

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.

Our algorithm requires some  $t \leq mn$  iterations. Let  $X$  denote the final rounded vector output by our algorithm. We now present the following three lemmas about our algorithm.

**Lemma 1** (i) *In any iteration of Phase 2, any floating machine has at most two floating jobs assigned fractionally to it.* (ii) *Let  $\phi$  and  $J'$  denote the fractional assignment and set of floating jobs respectively, at the beginning of Phase 2. Conditional on any values of these random variables, we have with probability one that for all  $i \in M$ ,  $\sum_{j \in J'} X_{i,j} \in \{ \lfloor \sum_{j \in J'} \phi_{i,j} \rfloor, \lceil \sum_{j \in J'} \phi_{i,j} \rceil \}$ .*

**Proof** We start by making some observations about the beginning of the first iteration of Phase 2. Consider the values  $v, m', n'$  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). ■

**Lemma 2** *For all  $i, j, h, \alpha$ ,  $\mathbf{E}[X_{i,j}^{(h+1)} \mid (X_{i,j}^{(h)} = \alpha)] = \alpha$ . In particular,  $\mathbf{E}[X_{i,j}^{(h)}] = x_{i,j}^*$  for all  $i, j, h$ .*

**Lemma 3** (i) *Let machine  $i$  be protected during iteration  $h + 1$ . Then  $\forall h' \in \{0, \dots, h + 1\}$ ,  $\sum_{j \in J} X_{i,j}^{(h')} \cdot p_{i,j} = \sum_{j \in J} x_{i,j}^* \cdot p_{i,j}$  with probability 1.* (ii) *For all  $i$ ,  $\sum_{j \in J} X_{i,j} \cdot p_{i,j} < \sum_{j \in J} x_{i,j}^* \cdot p_{i,j} + \max_{j \in J: x_{i,j}^* \in (0,1)} p_{i,j}$  with probability 1.*

**Proof** Part (i) follows from (1), and from the fact that if a machine is protected in any one iteration, it is also protected in all previous ones.

For part (ii), if  $i$  remained protected throughout the algorithm, then its total load never changes and the lemma holds. Assume  $i$  become a singleton machine when it became unprotected. The total load on  $i$  when it became unprotected is  $\sum_{j \in J} x_{i,j}^* \cdot p_{i,j}$  and irrespective of how the floating job on  $i$  gets rounded, the additional load on  $i$  is strictly less than  $\max_{j \in J: x_{i,j}^* \in (0,1)} p_{i,j}$ . Hence the lemma holds. Finally, assume that  $i$  had two floating jobs  $j_1$  and  $j_2$  when it became unprotected (Lemma 1(i) shows that this is the only remaining possibility); let the fractional assignments of  $j_1$  and  $j_2$  on  $i$  at this time be  $\phi_{i,j_1}$  and  $\phi_{i,j_2}$  respectively. Let  $\phi_{i,j_1} + \phi_{i,j_2} \in (0, 1]$ . Hence, by Lemma 1(ii), at most one of these jobs is finally assigned to  $i$ . So, the additional load on  $i$  is strictly less than  $\sum_{j \in J} x_{i,j}^* \cdot p_{i,j} + \max_{j \in J: x_{i,j}^* \in (0,1)} p_{i,j}$ . A similar argument holds when  $\phi_{i,j_1} + \phi_{i,j_2} \in (1, 2]$ . Hence, the lemma holds. ■

### 3 Weighted Completion Time and Makespan

We present a  $(\frac{3}{2}, 2)$ -bicriteria approximation algorithm for (weighted completion time, makespan) with unrelated parallel machines. 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 ideas of Skutella [19] and those of Section 2; 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 (see [19]): schedule the jobs in the order of non-increasing ratios  $\frac{w_j}{p_{i,j}}$ . Let this order on machine  $i$  be denoted  $\prec_i$ . Given an assignment-vector  $x$  and a machine  $i$ , let  $\Phi_i(x) = \sum_{(k,j): k \prec_i j} w_j x_{i,j} x_{i,k} p_{i,k}$ . 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

$$\left(\sum_{i,j} w_j p_{i,j} x_{i,j}\right) + \left(\sum_i \Phi_i(x)\right). \quad (2)$$

Given a pair  $(C, T)$ , we write the following Integer Quadratic Program (IQP) motivated by [19]. 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:

$$z \geq \left(\sum_j w_j \sum_i \frac{x_{i,j}(1+x_{i,j})}{2} p_{i,j}\right) + \left(\sum_i \Phi_i(x)\right); \quad (3)$$

$$z \geq \sum_j w_j \sum_i x_{i,j} p_{i,j}; \quad (4)$$

$$\forall i, T \geq \sum_j p_{i,j} x_{i,j}; \quad (5)$$

$$\forall(i, j), (p_{i,j} > T) \Rightarrow (x_{i,j} = 0). \quad (6)$$

The constraint (6) is easily seen to be valid, since we want solutions of makespan at most  $T$ . Next, since  $u(1+u)/2 = u$  for  $u \in \{0, 1\}$ , (2) shows that constraints (3) and (4) are valid:  $z$  denotes an upper bound on the weighted completion time, subject to the makespan being at most  $T$ . Crucially, as shown in [19], the quadratic constraint (3) 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  $\epsilon$  that is, say, any positive constant. As shown in [19], this is easily dealt with by derandomizing the algorithm. Let  $\epsilon$  be a suitably small positive constant. We find a (near-)optimal solution to the CPR, with additive error at most  $\epsilon$ . 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 employing our rounding algorithm of Section 2 on the fractional assignment  $x$ . Assuming that we obtained such a fractional assignment, let us now analyze this algorithm. Let  $X^{(h)}$  denote the (random) fractional assignment at the end of iteration  $h$  of our rounding algorithm. Our key lemma is:

**Lemma 4** 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  $h$ . Also fix the fractional assignment at the end of iteration  $h$  to be some arbitrary  $x^{(h)} = \{x_{i,j}^{(h)}\}$ . So, our goal is to show, conditional on this fractional assignment, that  $\mathbf{E}[\Phi_i(X^{(h+1)})] \leq \Phi_i(x^{(h)})$ . 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  $\alpha = \mathbf{E}[\Phi_i(X^{(h+1)})]/\Phi_i(x^{(h)})$  is maximized when all jobs with nonzero weight have the same  $\frac{w_j}{p_{i,j}}$  ratio. Partition the jobs into sets  $S_1, \dots, S_k$  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  $r_1, \dots, r_k$  be in non-decreasing order. For each job  $j \in S_1$ , we set  $w'_j = w_j + \lambda p_{i,j}$  where  $\lambda$  was

sufficiently small absolute value so that the relative ordering of  $r_1, \dots, r_k$  does not change. This changes the value of  $\alpha$  to a new value  $\alpha'(\lambda) = \frac{a+b\lambda}{c+d\lambda}$ , where  $a, b, c$  and  $d$  are constants independent of  $\lambda$ ,  $\alpha = a/c$ , and  $a, c > 0$ . Crucially, since  $\alpha'(\lambda)$  is a ratio of two linear functions, its value depends monotonically (either increasing or decreasing) on  $\lambda$ , in the allowed range for  $\lambda$ . Hence, there exists an allowed value for  $\lambda$  such that  $\alpha'(\lambda) \geq \alpha$ , and either  $r'_1 = r_2$  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 value  $\gamma > 0$ ,

$$\Phi_i(x^{(h)}) = \gamma \cdot \sum_{\{k,j\}:k < i} p_{i,j} p_{i,k} x_{i,j}^{(h)} x_{i,k}^{(h)};$$

$$\mathbf{E}[\Phi_i(X^{(h+1)})] = \gamma \cdot \mathbf{E}\left[\sum_{\{k,j\}:k < i} p_{i,j} p_{i,k} X_{i,j}^{(h+1)} X_{i,k}^{(h+1)}\right].$$

(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,  $\mathbf{E}[\Phi_i(X^{(h+1)})]$  equals

$$\begin{aligned} & \frac{\gamma}{2} \cdot \left(\mathbf{E}\left[\left(\sum_j p_{i,j} X_{i,j}^{(h+1)}\right)^2\right] - \sum_j \mathbf{E}\left[\left(p_{i,j} X_{i,j}^{(h+1)}\right)^2\right]\right) \\ & = \frac{\gamma}{2} \cdot \left(\left(\sum_j p_{i,j} x_{i,j}^{(h)}\right)^2 - \sum_j \mathbf{E}\left[\left(p_{i,j} X_{i,j}^{(h+1)}\right)^2\right]\right) \end{aligned}$$

where the latter equality follows since  $i$  is protected in iteration  $h + 1$ . Further, for any  $j$ , the probabilistic rounding ensures that there exists a pair of positive reals  $(u, v)$  such that  $\mathbf{E}\left[\left(X_{i,j}^{(h+1)}\right)^2\right] = \frac{v}{u+v} (x_{i,j}^{(h)} + u)^2 + \frac{u}{u+v} (x_{i,j}^{(h)} - v)^2 \geq (x_{i,j}^{(h)})^2$ . Hence,  $\mathbf{E}[\Phi_i(X^{(h+1)})] \leq \Phi_i(x^{(h)})$  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 the linearity of expectation.

**Case III:** Iteration  $h+1$  is in Phase 2, and  $i$  had two floating jobs then. (Lemma 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  $u, v$  such that we set  $(X_{i,j}^{(h+1)}, X_{i,j'}^{(h+1)})$  to  $(x_{i,j}^{(h)} + v, x_{i,j'}^{(h)} - v)$  with probability  $u/(u+v)$ , and to  $(x_{i,j}^{(h)} - u, x_{i,j'}^{(h)} + u)$  with probability  $v/(u+v)$ . We can verify now that  $\mathbf{E}[X_{i,j}^{(h+1)} \cdot X_{i,j'}^{(h+1)}] \leq x_{i,j}^{(h)} \cdot x_{i,j'}^{(h)}$ ; thus, the type (iii) term is also handled. ■

Lemma 4 leads to our main theorem here.

$$\sum_{i=1}^m t_i^p \leq T^p \quad (9)$$

**Theorem 5** *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  $(3C/2, 2T)$ .*

$$\sum_{i=1}^m \sum_{j=1}^n x_{i,j} \cdot p_{i,j}^p \leq T^p \quad (10)$$

$$\forall (i, j) \in \{1, \dots, m\} \times \{1, \dots, n\} \quad x_{i,j} \in \{0, 1\} \quad (11)$$

$$\forall (i, j) \in \{(i, j) \mid p_{i,j} > T\} \quad x_{i,j} = 0 \quad (12)$$

**Proof** For simplicity, we ignore the factor of  $\epsilon$ ; in the full version, we will show how it can be dealt with in the same simple manner as in [19]. The fact that  $T' \leq 2T$  with probability 1 easily follows by applying Lemma 3(ii) with constraints (5) and (6). Let us now bound  $\mathbf{E}[C']$ .

Recall that  $X = \{X_{i,j}\}$  denotes the final random integral assignment. Lemma 2 shows that  $\mathbf{E}[X_{i,j}] = x_{i,j}^*$ . Also, Lemma 4 shows that  $\mathbf{E}[\Phi_i(X)] \leq \Phi_i(x^*)$ , for all  $i$ . These, combined with the linearity of expectation, yield

$$\begin{aligned} \mathbf{E}\left[\left(\sum_j w_j \sum_i p_{i,j} X_{i,j}/2\right) + \left(\sum_i \Phi_i(X)\right)\right] &\leq \\ \left(\sum_j w_j \sum_i p_{i,j} x_{i,j}/2\right) + \left(\sum_i \Phi_i(x)\right) &\leq z \end{aligned}$$

where the second inequality follows from (3). Similarly, we have

$$\mathbf{E}\left[\sum_j w_j \sum_i X_{i,j} p_{i,j}\right] = \sum_j w_j \sum_i x_{i,j} p_{i,j} \leq z,$$

where the inequality follows from (4). As in [19], we get from (2) that  $\mathbf{E}[C'] = (\sum_{i,j} w_j p_{i,j} \mathbf{E}[X_{i,j}]) + (\sum_i \mathbf{E}[\Phi_i(X)]) = \mathbf{E}[(\sum_j w_j \sum_i p_{i,j} X_{i,j}/2) + (\sum_i \Phi_i(X))] + \mathbf{E}[\sum_j w_j \sum_i X_{i,j} p_{i,j}/2] \leq z + z/2 \leq 3C/2$ . 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 [5]. Let  $\{1, \dots, n\}$  and  $\{1, \dots, m\}$  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.

$$\forall j \in \{1, \dots, n\} \quad \sum_{i=1}^m x_{i,j} \geq 1 \quad (7)$$

$$\forall i \in \{1, \dots, m\} \quad \sum_{j=1}^n x_{i,j} \cdot p_{i,j} - t_i \leq 0 \quad (8)$$

We let  $x_{i,j} \geq 0$  for all  $(i, j)$  in the above integer program, to obtain a convex program. The feasibility of the convex program can be checked in polynomial time to within an additive error of  $\epsilon$  (for an arbitrary constant  $\epsilon > 0$ ): the nonlinear constraint (9) is not problematic since it defines a convex feasible region [5]. We obtain the minimum feasible value  $T^*$  of  $T$ , using bisection search. We ignore the additive error  $\epsilon$  in the rest of our discussions since our all our randomized guarantees can be obtained deterministically using the method of conditional probabilities in such a way that  $\epsilon$  is eliminated from the final cost. We also assume that  $T$  is set to  $T^*$ . We start with two lemmas involving useful calculations.

**Lemma 6** *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)  $O(\frac{2^p}{\sqrt{p}})$  if  $p$  sufficiently large. 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 7** *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 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, the ratio  $N/D$  is maximized when at least one of the variables  $a_1$  and  $a_2$  belongs to  $\{0, 1\}$ ; also, the maximum value of  $N/D$  is at most  $\gamma(p)$ , the value from Lemma 6.*

We once again round using our algorithm of Section 2, and analyze the rounding now. We now collect together some definitions that will be of use in Theorems 8 and 9.

**Some useful definitions.** We now recall a few definitions, and also define a few new ones.  $X$  denotes the final rounded assignment,  $\{x_{i,j}^*\}$  the fractional solution to the convex program,  $t_i^* = \sum_j p_{i,j} x_{i,j}^*$  is the fractional load on machine  $i$ , and  $T_i$  denotes the final (random) load on machine  $i$ . Let  $\mu_p(x, i) = \sum_j x_{i,j} p_{i,j}^p$  for any assignment-vector  $x$ ; we hope the two different occurrences of the symbol “ $p$ ” in “ $p_{i,j}^p$ ” do not cause confusion. We will sometimes fix the

machine  $i$ , and consider the situation where we currently have an assignment-vector  $x$ , with machine  $i$  being unprotected. In such a case, three definitions will be useful. (a) 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  $x$ . The cases where less than two jobs are floating on  $i$  are handled by introducing artificial new values  $j_1$  and/or  $j_2$  and setting one or both of the variables  $\{x_{i,j_1}, x_{i,j_2}\}$  to zero (or infinitesimally small); we do not consider these cases in the rest of our arguments. Note that  $j_1$  and  $j_2$  are functions of  $i$  and  $x$ , but we avoid explicitly writing so for the sake of conciseness. (b) Let  $R_i(x) = \sum_{j: x_{i,j}=1} p_{i,j}$  be the ‘‘already-rounded load’’ on  $i$  under assignment  $x$ . (c) Define  $\phi_p(x, i)$  to be: if  $x_{i,j_1} + x_{i,j_2} \in [0, 1]$ , then  $\phi_p(x, i) = (1 - x_{i,j_1} - x_{i,j_2}) \cdot R_i(x)^p + x_{i,j_1} \cdot (R_i(x) + p_{i,j_1})^p + x_{i,j_2} \cdot (R_i(x) + p_{i,j_2})^p$ ; else if  $x_{i,j_1} + x_{i,j_2} \in (1, 2]$ , then  $\phi_p(x, i) = (1 - x_{i,j_2}) \cdot (R_i(x) + p_{i,j_1})^p + (1 - x_{i,j_1}) \cdot (R_i(x) + p_{i,j_2})^p + (x_{i,j_1} + x_{i,j_2} - 1) \cdot (R_i(x) + p_{i,j_1} + p_{i,j_2})^p$ .

**Theorem 8** *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  $p$ . Further, for any fixed value of  $p > 2$  it is possible to achieve a better factor  $\rho(p)$  using numerical techniques. In particular, the following table illustrates certain achievable values of  $\rho(p)$ :*

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

**Sketch of Proof** Fix a machine  $i$ . If  $i$  was always protected, then  $T_i = t_i^*$ . Otherwise, let  $U = \{U_{i,j}\}$  denote the random fractional assignment at the beginning of the first iteration in which  $i$  became unprotected. Recall the definitions from the paragraph above, and in particular, those of  $j_1$  and  $j_2$ . By definition of a protected machine,  $R_i(U) + U_{i,j_1} \cdot p_{i,j_1} + U_{i,j_2} \cdot p_{i,j_2} = t_i^*$ . We claim that for any assignment-vector  $u$ ,  $\mathbf{E}[T_i^p | U = u] = \phi_p(u, i)$ . The reasoning is as follows. Recall that  $X = \{X_{i,j}\}$  denotes the final integral assignment. Then, conditional on ‘‘ $U = u$ ’’, we have: (i)  $X_{i,j_1} + X_{i,j_2} \leq 1$  with probability one since  $u_{i,j_1} + u_{i,j_2} \in [0, 1]$ ; and (ii)  $\mathbf{E}[X_{i,j_1}] = u_{i,j_1}$ ,  $\mathbf{E}[X_{i,j_2}] = u_{i,j_2}$ . In particular, we have

$$\Pr[X_{i,j_1} = X_{i,j_2} = 0] = 1 - u_{i,j_1} - u_{i,j_2}.$$

Using these observations, we get that  $\mathbf{E}[T_i^p | U = u] = \phi_p(u, i)$ . Similarly, in the case where  $u_{i,j_1} + u_{i,j_2} \in (1, 2]$ ,

we have  $X_{i,j_1} + X_{i,j_2} \geq 1$  with probability one; identical arguments show again that  $\mathbf{E}[T_i^p | U = u] = \phi_p(u, i)$ .

Recall the function  $\mu_p(x, i)$  from the paragraph on definitions preceding the statement of this theorem. Irrespective of the value of  $u_{i,j_1} + u_{i,j_2}$ , we have

$$\frac{\mathbf{E}[T_i^p | U = u]}{t_i^{*p} + \mathbf{E}[\mu_p(X, i) | U = u]} = \frac{\mathbf{E}[T_i^p | U = u]}{t_i^{*p} + \sum_j u_{i,j} p_{i,j}^p} \leq \frac{\mathbf{E}[T_i^p | U = u]}{t_i^{*p} + u_{i,j_1} p_{i,j_1}^p + u_{i,j_2} p_{i,j_2}^p} \leq \gamma(p).$$

The last inequality follows from Lemma 7. By rearranging this expression and unconditioning on the value of  $U$ , we get

$$\begin{aligned} \mathbf{E}[T_i^p] &\leq \gamma(p)(t_i^{*p} + \mathbf{E}[\mu_p(X, i)]) \\ &\leq \gamma(p)(t_i^{*p} + \sum_j x_{i,j}^* p_{i,j}^p) \text{ (by Lemma 2).} \end{aligned}$$

So,  $\sum_i \mathbf{E}[T_i^p] \leq 2\gamma(p) \cdot T^p$ , by (9) and (10). The claims for  $\rho(p)$  follow by noting that  $\rho(p) \leq (2\gamma(p))^{\frac{1}{p}}$  and substituting  $\gamma(p)$  from Lemma 6, and by the fact that for any non-negative random variable  $Z$ ,  $\mathbf{E}[Z] \leq (\mathbf{E}[Z^p])^{1/p}$ . ■

Note that the present proof of Theorem 8 basically uses the  $(1/2, 1/2)$ -convex combination of the constraints (9) and (10). In the full version of this paper, we will pursue (slightly) better approximation algorithms for the case of fixed  $p > 2$ , by considering other convex combinations.

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

We now present our multicriteria optimization results for a given collection of  $L_p$  norms and weighted completion time.

**Theorem 9** *Suppose we are given an instance of unrelated-machine scheduling, and a finite set of positive integer norms  $S$ . Suppose further that there exists an (unknown) schedule with  $L_p$  norm of machine-loads at most some given  $T(p)$  for each  $p \in S$ , and with weighted completion time at most some given  $W^*$ . Then, our rounding algorithm of Section 2 can be derandomized in polynomial time to guarantee any one of the following:*

1. For every  $p \in S$ , the rounded norm  $C(p) \leq 2.56 \cdot T(p)$ ;
2. The rounded completion time  $W(X) \leq 3.2 \cdot W^*$  and for every  $p \in S$ , the rounded norm  $C(p) \leq 3.2 \cdot T(p)$ ;
3. For any  $\epsilon > 0$ ,  $W(X) \leq \frac{3}{2} \cdot (1 + \epsilon)W^*$  and for every  $p \in S$ ,  $C(p) \leq 2(e + \frac{2}{\epsilon}) \cdot T(p)$ , where  $e$  is the base of natural logarithms;
4. There exists a constant  $K$  such that if  $S = \{p\}$ , then for any  $\epsilon > 0$  and any  $p \geq \frac{K}{\epsilon^2}$ ,  $W(X) \leq \frac{3}{2}(1 + \epsilon)$  and  $C(p) \leq 2 \cdot T(p)$ .

**Sketch of Proof** We only discuss the main arguments for proving guarantee 1 of the Lemma. Specifically, given a collection of integer norms  $S$  and a target  $L_p$  norm  $T(p)$  for each  $p \in S$ , we either prove that no assignment exists which simultaneously satisfies all these targets or obtain an integral assignment where the final  $L_p$  norm for any  $p \in S$  is at most  $2.56T(p)$ . The convex program we use is a variant of the one in Section 4. Instead of the constraints (9) and (10), we have two such constraints for each  $p \in S$ :  $\sum_{i=1}^m t_i^p \leq T(p)^p$  and  $\sum_{i=1}^m \sum_{j=1}^n x_{i,j} \cdot p_{i,j}^p \leq T(p)^p$ .

If the above convex program is infeasible, then we can clearly declare that no valid assignment exists which respects the targets. Assume that the convex program is feasible and  $x^*$  is the feasible fractional assignment. We will describe a derandomization of the algorithm in Section 2, in order to get the guarantee for all  $p \in S$ . Recall the definitions from the paragraph preceding Theorem 8. Let  $X^{(h)}$  denote the (fractional) assignment vector after iteration  $h$  in our derandomized rounding algorithm; so,  $X^{(0)} \doteq x^*$ . Given a fractional assignment  $x$  that occurs in our algorithm (i.e.,  $x = X^{(h)}$  for some  $h$ ) in which machine  $i$  is unprotected, we define  $\psi_p(x, i)$  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 7 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:

$$\psi_p(x, i) \leq \mu_p(x, i). \quad (13)$$

Let  $M_1^{(h)}$  and  $M_2^{(h)}$  denote the set of protected and unprotected machines respectively immediately after iteration  $h$ . Let  $Q_p(X^{(h)}) = \sum_{i \in M_1^{(h)}} \mu_p(X^{(h)}, i) + \sum_{i \in M_2^{(h)}} \psi_p(X^{(h)}, i)$ . We now define the ‘‘potential function’’ for our derandomization,

$$Q(X^{(h)}) = \sum_{p \in S} \frac{Q_p(X^{(h)})}{f(p) \cdot T(p)^p},$$

where the positive values  $f(p)$  are chosen such that

$$\sum_{p \in S} \frac{1}{f(p)} \leq 1.$$

Note that

$$Q(X^{(0)}) = \sum_{p \in S} \frac{\sum_i \mu_p(X^{(0)}, i)}{f(p) \cdot T(p)^p} \leq 1,$$

where the inequality follows from (10). Our aim is to use the method of conditional probabilities to ensure that at the end, we have  $Q(X) \leq 1$ .

We are now ready to describe the derandomized version of our rounding algorithm. In iteration  $h + 1$ , as in

the randomized version, we have two choices of assignment vectors  $x_1$  and  $x_2$  and two scalars  $\alpha_1, \alpha_2 \geq 0$  and  $\alpha_1 + \alpha_2 = 1$  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  $x$ . Next, if a machine  $i$  becomes unprotected at the end of this  $(h + 1)^{\text{st}}$  iteration, then for all  $p \in S$ , we need to replace  $\mu_p(X^{(h+1)}, i)$  by  $\psi_p(X^{(h+1)}, i)$  in the expression for  $Q$ . It follows from (13) 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  $C(1) = \sum_i \phi_1(X, i) = Q_1(X) \leq f(1)T(1)$ . We now analyze the value  $C(p)$  for norms  $p > 1$ . Suppose the algorithm terminates after  $\ell$  iterations. (So,  $X = X^{(\ell)}$ .) We have

$$\sum_{i \in M_1^{(\ell)}} T_i^p = \sum_{i \in M_1^{(\ell)}} t_i^{*p}.$$

Next, a moment’s reflection shows that for any  $i \in M_2^{(\ell)}$ ,  $T_i^p = \phi_p(X, i)$ . So,  $\sum_{i \in M_2^{(\ell)}} T_i^p = \sum_{i \in M_2^{(\ell)}} \phi_p(X, i) = \sum_{i \in M_2^{(\ell)}} \gamma(p)(\psi_p(X, i) + t_i^{*p}) \leq \gamma(p)(Q_p(X) + \sum_{i \in M_2^{(\ell)}} t_i^{*p}) \leq \gamma(p)(f(p)T(p)^p + \sum_{i \in M_2^{(\ell)}} t_i^{*p})$ . Thus, letting ‘‘ $\sum_i$ ’’ denote the sum over all the machines, we get that  $C(p)^p = \sum_i T_i^p$  is at most

$$\gamma(p)(f(p)T(p)^p + \sum_i t_i^{*p}) \leq \gamma(p)(f(p) + 1)T(p)^p.$$

Hence,  $C_p \leq (\gamma(p)(1 + f(p)))^{\frac{1}{p}} T(p)$ . We are now left to show the choice of values  $f(p)$  such that  $f(1) = 2.56$ ,  $(\gamma(p)(1 + f(p)))^{\frac{1}{p}} \leq 2.56$  for all integers  $p > 1$ , and  $\sum_{p=1}^{\infty} \frac{1}{f(p)} \leq 1$ , where the summation is over the set of positive integers. 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 6, 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  $\sum_{p=1}^{\infty} \frac{1}{f(p)} \leq 1$ . ■

The *restricted assignment* case of unrelated-machine scheduling is where for each  $j$ , there is a value  $p_j$  such that for all  $i$ ,  $p_{i,j} \in \{p_j, \infty\}$ . The following theorem pertains to the approximation ratio of our algorithm in Section 2 for the restricted assignment case. As in [6], we first obtain the unique fractional solution  $x^*$  which is *simultaneously* optimal with respect all norms  $p \geq 1$ . Azar *et al.* [6]

show that  $x^*$  can be rounded efficiently to get an *absolute* 2-approximation factor w.r.t. every norm  $p \geq 1$ . (That is, each  $L_p$  norm is individually at most twice optimal). This result was also independently shown by [10]. We get an improvement as follows:

**Theorem 10** *Given an all-norm fractionally optimal assignment  $x^*$ , and a fixed norm  $p' \in [1, \infty)$ , our rounding algorithm can be derandomized in polytime 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  $p > 1$ , where  $\rho(p')$  is the function from Theorem 8.*

## 6 Generalizing the Karp *et al.* procedure and applications

We now employ some of the ideas behind our algorithm of Section 2 to develop two additional applications. The 2-approximation for makespan minimization in unrelated parallel machines [15] was extended in [18] 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 as well as the constraint  $\sum_{i,j} c_{i,j} x_{i,j} = C$  for some  $C$ . Then, the algorithm of [18] constructs an integral assignment  $y$  such that  $\sum_j p_{i,j} y_{i,j} \leq 2T$  for all  $i$ , and  $\sum_{i,j} c_{i,j} y_{i,j} \leq C$ . We now describe how a basic rounding theorem of Karp *et al.* [12] can be used to obtain the result of [15]. We then show a probabilistic generalization of this theorem of [12] (Theorem 11) which yields the result of [18]. We also describe an extension (Theorem 12) 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. In the setting of [12], we are given a matrix  $A \in \mathbb{R}^{m \times n}$ , with  $t$  denoting  $\max_j \{\sum_{i:A_{ij}>0} A_{ij}, -\sum_{i:A_{ij}<0} A_{ij}\}$ . Then, it is shown in [12] that for any given real vector  $x = (x_1, x_2, \dots, x_n)^T$ , we can efficiently find a *rounded* counterpart  $y = (y_1, y_2, \dots, y_n)^T$  such that  $\|Ay - Ax\|_\infty < t$ . To see how this yields the result of [15], first, consider the standard LP: **(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 sense of the Karp *et al.* result [12], can be taken to be  $T$ , and therefore there is an integral vector  $y$  such that: (i) for each  $j$ ,  $\sum_i -y_{i,j} < 0$ , or  $\sum_j y_{i,j} \geq 1$  (i.e., job  $j$  is assigned to some machine), and (ii) for each  $i$ ,  $\sum_j p_{i,j} y_{i,j} \leq 2T$ . We now describe our probabilistic generalization of the theorem of [12]:

**Theorem 11** *Given a matrix  $A \in \mathbb{R}^{m \times n}$  with  $t$  denoting  $\max_j \{\sum_{i:A_{ij}>0} A_{ij}, -\sum_{i:A_{ij}<0} A_{ij}\}$ , and a frac-*

*tional vector  $x$ , we can in randomized polynomial time construct a vector  $y$  such that: (i)  $\forall j, y_j \in \{\lfloor x_j \rfloor, \lceil x_j \rceil\}$  with probability 1, (ii)  $\forall i, (Ay)_i < (Ax)_i + t$ , with probability 1, and (iii) for each  $j$ ,  $\mathbf{E}[y_j] = x_j$ . In particular, given any vector  $\vec{c}$ , we have  $\mathbf{E}[\sum_j c_j y_j] = C \doteq \sum_j c_j x_j$ . Furthermore, a vector  $y$  for which (i) and (ii) hold, and for which  $\sum_j c_j y_j \leq C$ , can be constructed in deterministic polynomial time.*

**Proof** (Sketch) The result of [12] describes a deterministic algorithm (denoted by  $\mathcal{A}$ ) to obtain a rounded vector  $y$ , satisfying the properties (i) and (ii) of this theorem. We first recap the main ideas underlying the result of [12]. By subtracting out integer parts, we may assume that  $x_j \in [0, 1]$  for all  $j$ . At any point in  $\mathcal{A}$ , once we round  $x_j$  to 0 or 1, we will never alter it. We keep rounding variables incrementally; suppose  $S$  is the current set of indices  $j$  such that  $x_j \in (0, 1)$ . Let  $|S| = s$ . Thus,  $x$  is a point in  $(0, 1)^s$ , and we will modify  $x$  so that at least one more variable gets rounded, as follows. Consider the linear system restricted to the set of variables in  $S$ . It is shown in [12] that: (a) either this system is under-determined, or (b) there exists a row  $i$  (which can be found efficiently) such that no matter how we round the variables in  $S$  to get a final vector  $y$ , we will have  $(Ay)_i < b_i + t$ . If (b) holds, we can keep discarding such rows  $i$  from consideration (since they are “safe” from now on), until case (a) holds. Now, if case (a) holds, elementary linear algebra shows that there is a direction  $\vec{r}$  that we can follow starting from the current point  $x \in (0, 1)^s$ , such that all values  $(Ax)_i$  remain unchanged. Thus, the approach of [12] is to travel along  $\vec{r}$  starting from  $x$ , until we hit a face of the  $s$ -dimensional cube. The result of [12] follows by repeating the rounding in this manner.

We now randomize the rounding algorithm  $\mathcal{A}$  as follows. Note that there are two opposite directions in which we can travel along  $\vec{r}$  starting from  $x$ :  $\vec{r}$  and  $-\vec{r}$ . Let  $\alpha$  and  $\beta$  be the positive quantities such that the points  $p_1 = x + \alpha\vec{r}$  and  $p_2 = x - \beta\vec{r}$  lie on a face of the  $s$ -dimensional cube. Then, we choose to move to  $p_1$  with probability  $\beta/(\alpha + \beta)$ , and to  $p_2$  with the complementary probability of  $\alpha/(\alpha + \beta)$ . We can check that **(B1)** and **(B2)** continue to hold with probability one at the end of this algorithm. The added advantage of our method is that due to our random choice, the expected change in any dimension of our vector  $x$  is zero; an easy induction over time then establishes property (ii) of the statement of Theorem 11. Finally, property (iii) can be established by derandomizing this algorithm using the method of conditional probabilities. ■

Theorem 11 can be similarly seen to imply the result of [18]. We also extend Theorem 11 in the following useful manner: suppose, in addition to the system  $Ax \leq b$ , we have  $\ell$  additional constraints  $c^{(k)} \cdot x \leq d_k, k = 1, \dots, \ell$ . Let  $M_k$  be the maximum absolute value of any component in

$c^{(k)}$ , and  $t$  be as before. Let  $\epsilon > 0$  be any parameter. We show:

**Theorem 12** *Given a system  $Ax = b$  as in the setting of [18], with  $\ell$  additional linear constraints, we can, in randomized polynomial time, construct a random vector  $y$  such that: (i)  $\forall j, y_j \in \{\lfloor x_j \rfloor, \lceil x_j \rceil\}$ ; (ii) for all row indices  $i$  in  $A$ ,  $(Ay)_i < b_i + t(1 + \epsilon)$ ; (iii) for each  $j$ ,  $\mathbf{E}[y_j] = x_j$ ; (iv) for each of the additional constraints  $c^{(k)} \cdot x \leq d_k, k = 1, \dots, \ell$ , we have  $|c^{(k)} \cdot y - d_k| = O(M_k \ell / \epsilon)$ , where  $M_k$  is the maximum absolute value of any component in  $c^{(k)}$ .*

Note in particular that for bounded  $M_k, \ell$  and  $\epsilon$ , 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  $\ell$ .

We finally consider the problem of unrelated parallel-machine scheduling with resource-dependent processing times. This is a generalization 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 algorithm for minimizing makespan in this setting. A direct application of Theorem 11 yields an assignment of jobs and resources to machines; combined with the resource-scheduling algorithm of [11], we get a 4-approximation for this problem.

**Theorem 13** *There exists a polynomial-time 4-approximation algorithm for the problem of minimizing makespan in unrelated parallel machine scheduling with resource-dependent processing times.*

We will present the details in the full version.

**Acknowledgments.** We thank David Shmoys for valuable discussions, Cliff Stein for introducing us to [20], and Yossi Azar for sending us an early version of [5]. We are thankful to the anonymous referees for valuable comments.

## References

- [1] N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 493–500, 1997.
- [2] A. Archer. *Mechanisms for Discrete Optimization with Rational Agents*. PhD thesis, Cornell University, Jan 2004.
- [3] J. Aslam, A. Rasala, C. Stein, and N. Young. Improved bicriteria existence theorems for scheduling. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 846–847, 1999.
- [4] B. Awerbuch, Y. Azar, E. F. Grove, M.-Y. Kao, P. Krishnan, and J. S. Vitter. Load balancing in the  $L_p$  norm. In *IEEE Symposium on Foundations of Computer Science*, pages 383–391, 1995.
- [5] Y. Azar and A. Epstein. Convex programming for scheduling unrelated parallel machines. In *Proc. of the ACM Symposium on Theory of Computing*, pages 331–337, 2005.
- [6] Y. Azar, L. Epstein, Y. Richter, and G. J. Woeginger. All-norm approximation algorithms. *J. Algorithms*, 52(2):120–133, 2004.
- [7] Y. Azar and S. Taub. All-norm approximation for scheduling on identical machines. In *SWAT*, pages 298–310, 2004.
- [8] A. K. Chandra and C. K. Wong. Worst-case analysis of a placement algorithm related to storage allocation. *SIAM J. on Computing*, 4(3):249–263, 1975.
- [9] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding in bipartite graphs. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 323–332, 2002.
- [10] A. Goel and A. Meyerson. Simultaneous optimization via approximate majorization for concave profits or convex costs. Tech. Report CMU-CS-02-203, December 2002, Carnegie-Mellon University.
- [11] A. Grigoriev, M. Sviridenko, and M. Uetz. Unrelated parallel machine scheduling with resource dependent processing times. In *Integer Programming and Combinatorial Optimization (IPCO)*, pages 182–195, 2005.
- [12] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. V. Vazirani, and V. V. Vazirani. Global wire routing in two-dimensional arrays. *Algorithmica*, pages 113–129, 1987.
- [13] J. Kleinberg, E. Tardos, and Y. Rabani. Fairness in routing and load balancing. *J. Comput. Syst. Sci.*, 63(1):2–20, 2001.
- [14] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *Sequencing and scheduling: algorithms and complexity*. Elsevier, 1993.
- [15] J. K. Lenstra, D. B. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, pages 259–271, 1990.
- [16] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, pages 365–374, 1987.
- [17] P. Schuurman and G. J. Woeginger. Polynomial time approximation algorithms for machine scheduling: Ten open problems. *J. Scheduling*, pages 203–213, 1999.
- [18] D. B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, pages 461–474, 1993.
- [19] M. Skutella. Convex quadratic and semidefinite relaxations in scheduling. *Journal of the ACM*, 46(2):206–242, 2001.
- [20] C. Stein and J. Wein. On the existence of schedules that are near-optimal for both makespan and total weighted completion time. *Operations Research Letters*, 21, 1997.