A NEW APPROXIMATION TECHNIQUE FOR RESOURCE-ALLOCATION PROBLEMS

BARNA SAHA* AND ARAVIND SRINIVASAN †

Abstract. We develop a rounding method based on random walks in polytopes, which leads to improved approximation algorithms and integrality gaps for several assignment problems that arise in resource allocation and scheduling. In particular, it generalizes the work of Shmoys & Tardos on the generalized assignment problem in two different directions, where the machines have hard capacities, and where some jobs can be dropped. We also outline possible applications and connections of this methodology to discrepancy theory and iterated rounding.

Key words. Scheduling; rounding; approximation algorithms; integrality gap

1. Introduction. The "relax-and-round" paradigm is a well-known approach in combinatorial optimization. Given an instance of an optimization problem, we *enlarge* the set of feasible solutions I to some set $I' \supset I$ – often the linear-programming (LP) relaxation of the problem; we then map an (efficiently computed, optimal) solution $x^* \in I'$ to some "nearby" $x \in I$ and prove that x is near-optimal in I. This second "rounding" step is often a crucial ingredient, and many general techniques have been developed for it. In this work, we present a new rounding methodology which leads to several improved approximation algorithms in scheduling, and which, as we explain, appears to have connections and applications to other techniques and problems, respectively.

We next present background on (randomized) rounding and a fundamental scheduling problem, before describing our contribution.

Our work generalizes various *dependent* randomized rounding techniques that have been developed over the past decade or so. Recall that in randomized rounding, we use randomization to map $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ back to some $x = (x_1, x_2, \ldots, x_n)$ [41]. Typically, we choose a value α that is problem-specific, and, *independently* for each i, define x_i to be 1 with probability αx_i^*, and to be 0 with the complementary probability of $1 - \alpha x_i^*$. Independence can, however, lead to noticeable deviations from the mean for random variables that are *required* to be very close to (or even be equal to) their mean. A fruitful idea developed in [47, 29, 34] is to carefully introduce *dependencies* into the rounding process: in particular, some sums of random variables are held fixed with probability one, while still retaining randomness in the individual variables and guaranteeing certain types of negative-correlation properties among them. See [1] for a related deterministic approach that precedes these works. These dependent-rounding approaches lead to numerous improved approximation algorithms in scheduling, packet-routing and in several problems of combinatorial optimization [1, 47, 29, 34, 18].

We now introduce a fundamental scheduling model, which has spurred many advances and applications in combinatorial optimization, including linear-, quadratic- & convex-programming relaxations and new rounding approaches [36, 43, 45, 6, 22,

*Dept. of Computer Science, University of Maryland, College Park, MD 20742. Research supported by NSF Award NSF CCF-0728839. (barna@cs.umd.edu).

†Dept. of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Supported in part by NSF ITR Award CNS-0426683 and NSF Award CNS-0626636.(srin@cs.umd.edu)

A preliminary version of this paper appeared in the conference proceedings of Innovations in Computer Science (ICS) 2010.

34, 31, 8, 10, 15]. This model, *scheduling with unrelated parallel machines* (UPM) – and its relatives – play a key role in this work. Herein, we are given a set J of n jobs, a set M of m machines, and non-negative values $p_{i,j}$ ($i \in M,\ j \in J$): each job j has to be assigned to some machine, and assigning it to machine i will impose a processing time of $p_{i,j}$ on machine i. (The word "unrelated" arises from the fact that there may be no pattern among the given numbers $p_{i,j}$.) Variants such as the type of objective function(s) to be optimized in such an assignment, whether there is an additional "cost-function", whether a few jobs can be dropped, and situations where there are release dates for, and precedence constraints among, the jobs, lead to a rich spectrum of problems and techniques. We now briefly discuss two such highly-impactful results [36, 43]. The primary UPM objective in these works is to minimize the *makespan* – the maximum total load on any machine. It is shown in [36] that this problem can be approximated to within a factor of 2; furthermore, even some natural special cases cannot be approximated better than 1.5 unless $P = NP$ [36]. Despite much effort, these bounds have not been improved. The work of [43] builds on the upper-bound of [36] to consider the *generalized assignment problem* (GAP) where we incur a cost $c_{i,j}$ if we schedule job j on machine i; a simultaneous $(2, 1)$–approximation for the (makespan, total cost)-pair is developed in [43], leading to numerous applications (see, e.g., [2, 17]).

We generalize the methods of [33, 1, 47, 29, 34], via a type of random walk toward a vertex of the underlying polytope that we outline next. We then present several applications in scheduling and bipartite matching through problem-specific specializations of this approach, and discuss further prospects for this methodology.

The rounding approaches of [33, 1, 47, 29] are generalized to linear systems as follows in [34]. Suppose we have an n-dimensional constraint system $Ax \leq b$ with the additional constraints that $x \in [0, 1]^n$. This will often be an LP-relaxation, which we aim to round to some $y \in \{0, 1\}^n$ such that some constraints in "$Ay \leq b$" hold with probability one, while the rest are violated "a little" (with high probability). Given some $x \in [0, 1]^n$, the rounding approach of [34] is as follows. First, we assume without loss of generality that $x \in (0, 1)^n$: those x_j that get rounded to 0 or 1 at some point, are held fixed from then on. Next, we "judiciously" drop some of the constraints in "$Ax \leq b$" until the number of constraints becomes smaller than n, thus making the system linearly-dependent – leading to the efficient computation of an $r \in \Re^n$ that is in the nullspace of this reduced system. We then compute positive scalars α and β such that $x_1 := x + \alpha r$ and $x_2 := x - \beta r$ both lie in $[0, 1]^n$, and both have at least one component lying in $\{0, 1\}$; we then update x to a random Y as: $Y := x_1$ with probability $\beta/(\alpha + \beta)$, and $Y := x_2$ with the complementary probability $\alpha/(\alpha + \beta)$. Thus we have rounded at least one further component of x, and also have the useful property that for all j, $\mathbf{E}[Y_j] = x_j$. Different ways of conducting the "judicious" reduction lead to a variety of improved scheduling algorithms in [34]. The setting of [47, 29] on bipartite b-matchings can be interpreted in this framework.

We further generalize the above-sketched approach of [34]. Suppose we are given a polytope $\mathcal{P}$ in n dimensions, and a *non-vertex* point x belonging to $\mathcal{P}$. An appropriate basic-feasible solution will of course lead us to a vertex of $\mathcal{P}$, but we approach (not necessarily reach) a vertex of $\mathcal{P}$ by a random walk as follows. Let $\mathcal{C}$ denote the set of constraints defining $\mathcal{P}$ which are satisfied *tightly* (i.e., with equality) by x. Then, note that there is a non-empty linear subspace S of $\Re^n$ such that for any nonzero $r \in S$, we can travel up to some strictly-positive distance $f(r)$ along r starting from x, while staying in $\mathcal{P}$ and *continuing* to satisfy all constraints in $\mathcal{C}$ tightly. Our broad approach

to conduct a random move $Y := x + R$ by choosing an appropriately random R from S, such that the property "$\mathbf{E}[Y_j] = x_j$" of the previous paragraph still holds. In particular, let **RandMove**$(x, \mathcal{P})$ – or simply **RandMove**(x) if $\mathcal{P}$ is understood – be as follows. Choose a nonzero $r \in S$ arbitrarily, and set $Y := x + f(r)r$ with probability $f(-r)/(f(r) + f(-r))$, and $Y := x - f(-r)r$ with the complementary probability of $f(r)/(f(r) + f(-r))$. Note that if we repeat **RandMove**, we obtain a random walk that finally leads us to a vertex of $\mathcal{P}$; the high-level idea is to intersperse this walk with the idea of "judiciously dropping some constraints" from the previous paragraph, as well as combining certain constraints together into one. Three major differences from [34] are: (a) the care given to the tight constraints $\mathcal{C}$, (b) the choice of which constraint to drop being based on $\mathcal{C}$, and (c) clubbing some constraints into one. As discussed next, this recipe appears fruitful in a number of directions in scheduling, and as a new rounding technique in general.

Capacity constraints on machines, random matchings with sharp tail bounds. Handling "hard capacities" – those that cannot be violated – is generally tricky in various settings, including facility-location and other covering problems [21, 27, 38]. Motivated by problems in crew-scheduling [23, 42] and by the fact that servers have a limit on how many jobs can be assigned to them, the natural question of scheduling with a hard capacity-constraint of "at most b_i jobs to be scheduled on each machine i" has been studied in [50, 54, 53, 52, 19]. Most recently, the work of [19] has shown that this problem can be approximated to within a factor of 3 in the special case where the machines are *identical* (job j has processing time p_j on any machine). In § 2, we use our random-walk approach to generalize this to the setting of GAP and obtain the GAP bounds of [43] – i.e., approximation ratios of 2 and 1 for the makespan and cost respectively, while satisfying the capacity constraints: the improvements are in the more-general scheduling model, handling the cost constraint, and in the approximation ratio. We anticipate that such a capacity-sensitive generalization of [43] would lead to improved approximation algorithms for several applications of GAP, and present one such in Section 5.

Theorem 2.1 generalizes such capacitated problems to random bipartite b-matchings with target degree bounds and sharp tail bounds for given linear functions; see [24] for applications to models for complex networks. Recall that a (b)-matching is a subgraph in which every vertex v has degree at most $b(v)$. Given a *fractional* b-matching x in a bipartite graph $G = (J, M, E)$ of N vertices and a collection of k linear functions $\{f_i\}$ of x, many works have considered the problem of constructing (b-)matchings X such that $f_i(X)$ is "close" to $f_i(x)$ simultaneously for each i [3, 30, 40, 29]. The works [30, 40] focus on the case of constant k; those of [3, 29] consider general k, and require the usual "discrepancy" term of $\Omega(\sqrt{f_i(x) \log N})$ in $|f_i(X) - f_i(x)|$ for most/all i; in a few cases, $o(N)$ vertices will have to remain unmatched also. In contrast, Theorem 2.1 shows that if there is one structured objective function f_i with bounded coefficients associated with each $i \in M$, then in fact all the $|f_i(X) - f_i(x)|$ can be bounded independent of N. This appears to be the first such result here, and helps with equitable max-min fair allocations as discussed below.

Scheduling with outliers: makespan and fairness. Note that the $(2, 1)$ bicriteria approximation that we obtain for GAP above, generalizes the results of [43]. We now present such a generalization in another direction: that of "outliers" in scheduling [31]. For instance, suppose in the "processing times $p_{i,j}$ and costs $c_{i,j}$" setting of GAP, we also have a profit π_j for choosing to schedule each job j. Given a "hard" target profit Π, target makespan T and total cost C, the LP-rounding method of [31]

either proves that these targets are not simultaneously achievable, or constructs a schedule with values $(\Pi, 3T, C(1+\epsilon))$ for any constant $\epsilon > 0$. Using our rounding approach, we improve this to $(\Pi, (2+\epsilon)T, C(1+\epsilon))$ in § 3. (The factors of ϵ in the cost are required due to the hardness of knapsack [31].) Also, fairness is a fundamental issue in dealing with outliers: e.g., in repeated runs of such algorithms, we may not desire long starvation of individual job(s) in sacrifice to a global objective function. Theorem 3.2 accommodates fairness in the form of scheduling-probabilities for the jobs that can be part of the input.

Max-Min Fair Allocation. This problem is the max-min version of UPM, where we aim to maximize the minimum "load" (viewed as utility) on the machines; it has received a good deal of attention recently [8, 5, 25, 4, 10, 15]. We are able to employ a generalization of dependent randomized rounding to near-optimally determine the integrality gap of a well-studied LP relaxation. Also, Theorem 2.1 lets us generalize a result of [14] on max-min fairness to the setting of equitable partitioning of the jobs; see § 4.

Directions for the future: some potential connections and applications. Distributions on *structured* matchings in bipartite graphs is a topic that models many scenarios in discrete optimization, and we view our work as a useful contribution to it. We explore further applications and connections in § 6. A general question involving "rounding well" is the *lattice approximation* problem [41]: given $A \in \{0,1\}^{m \times n}$ and $p \in [0,1]^n$, we want a $q \in \{0,1\}^n$ such that $\|A \cdot (q-p)\|_\infty$ is "small"; the linear discrepancy of A is defined to be $\text{lindisc}(A) = \max_{p \in [0,1]^n} \min_{q \in \{0,1\}^n} \|A \cdot (q-p)\|_\infty$. The field of *combinatorial discrepancy theory* [13] has developed several classical results that bound $\text{lindisc}(A)$ for various matrix families A; column-sparse matrices have received much attention in this regard. Section 6 discusses a concrete approach to use our method for the famous Beck-Fiala conjecture on the discrepancy of column-sparse matrices [12], in the setting of random matrices. § 6 also suggests that there may be deeper connections to *iterated rounding*, a fruitful approach in approximation algorithms [32, 26, 44, 35, 51]. We view our approach as having broader connections/applications (e.g., to open problems including capacitated facility location [38]), and are studying these directions.

2. Random Matchings with Linear Constraints, and GAP with Capacity Constraints. We develop an efficient scheme to generate random subgraphs of bipartite graphs that satisfy hard degree-constraints and near-optimally satisfy a collection of linear constraints:

THEOREM 2.1. *Let $G = (J, M, E)$ be a bipartite graph with "jobs" J and "machines" M. Let $\mathcal{F}$ be the collection of edge-indexed vectors y (with $y_{i,j}$ denoting y_e where $e = (i,j) \in E$). Suppose we are given: (i) an integer* requirement *r_j for each $j \in J$ and an integer* capacity *b_i for each $i \in M$; (ii) for each $i \in M$, a linear objective function $f_i : \mathcal{F} \to \Re$ given by $f_i(y) = \sum_{j:\ (i,j) \in E} p_{i,j} y_{i,j}$ such that $0 \le p_{i,j} \le \ell_i$ for each j, (iii) a global cost constraint $\sum_{i,j} c_{i,j} y_{i,j} \le C$, and (iv) a vector $x \in \mathcal{F}$ with $x_e \in [0,1]$ for each e. Then, we can efficiently construct a random subgraph of G given by a binary vector $X \in \mathcal{F}$, such that: (a) with probability one, each $j \in J$ has degree at least r_j, each $i \in M$ has degree at most b_i, and $|f_i(X) - f_i(x)| < \ell_i \ \forall i$; and (b) for all $e \in E$, $\mathsf{E}[X_e] = x_e$ which implies $\mathsf{E}\left[\sum_{i,j} c_e X_e\right] = \sum_e c_e x_e = C$*

We first prove an important special case of Theorem 2.1: GAP with individual capacity constraints on each machine. This special case, handled by Theorem 2.2 captures much of the essence of Theorem 2.1; the full proof of Theorem 2.1 follows after Theorem 2.2.

The capacity constraint specifies the maximum number of jobs that can be scheduled on any machine, and is a hard constraint. Formally the problem is as follows, where $x_{i,j}$ is the indicator variable for job j being scheduled on machine i. Given m machines and n jobs, where job j requires a processing time of $p_{i,j}$ in machine i and incurs a cost of $c_{i,j}$ if assigned to i, the goal is to minimize the makespan $T = \max_i \sum_j x_{i,j} p_{i,j}$, subject to the constraint that the total cost $\sum_{i,j} x_{i,j} c_{i,j}$ is at most C and for each machine i, $\sum_j x_{i,j} \leq b_i$. C is the given upper bound on total cost and b_i is the capacity of machine i, that must be obeyed.

Our main contribution here is an efficient algorithm **Sched-Cap** that has the following guarantee, generalizing the GAP bounds of [43]:

THEOREM 2.2. *There is an efficient algorithm **Sched-Cap** that returns a schedule maintaining all the capacity constraints, of cost at most C and makespan at most $2T$, where T is the optimal makespan with cost C that satisfies the capacity constraints.*

Algorithm Sched-Cap. Algorithm **Sched-Cap** proceeds as follows. First we guess the optimum makespan T by binary search as in [36]. If $p_{i,j} > T$, $x_{i,j}$ is set to 0. The solution to the following integer program gives the optimum schedule:

$$\sum_{i,j} c_{i,j} x_{i,j} \leq C \qquad \text{(Cost)}$$

$$\sum_{i,j} x_{i,j} = 1 \ \forall j \qquad \text{(Assign)}$$

$$\sum_{j} p_{i,j} x_{i,j} \leq T \ \forall i \qquad \text{(Load)}$$

$$\sum_{j} x_{i,j} \leq b_i \ \forall i \qquad \text{(Capacity)}$$

$$x_{i,j} \in \{0,1\} \ \forall i,j$$

$$x_{i,j} = 0 \quad \text{if } p_{i,j} > T$$

We relax the constraint "$x_{i,j} \in \{0,1\}\ \forall (i,j)$" to "$x_{i,j} \in [0,1]\ \forall (i,j)$" to obtain an LP relaxation **LP-Cap**. We solve the LP to obtain an optimal LP solution x^*; we next show how **Sched-Cap** rounds x^* to obtain an integral solution within the approximation guarantee.

Note that $x^*_{i,j} \in [0,1]$ denotes the "fraction" of job j assigned to machine i. Initialize $X = x^*$. The algorithm is composed of several iterations. The random value of the assignment-vector X at the end of iteration h of the overall algorithm is denoted by X^h. Each iteration h conducts a randomized update using the **RandMove** on the polytope of a linear system constructed from a *subset* of the constraints of **LP-Cap**. Therefore, by induction on h, we will have for all (i,j,h) that $\mathsf{E}\left[X^h_{i,j}\right] = x^*_{i,j}$;*we use this property and drop the cost constraint since on expectation it is maintained.*

Let J and M denote the set of jobs and machines, respectively. Suppose we are at the beginning of some iteration $(h+1)$ of the overall algorithm: we are currently looking at the values $X^h_{i,j}$. We will maintain four invariants.

Invariants across iterations:

(I1) Once a variable $x_{i,j}$ gets assigned to 0 or 1, it is never changed;
(I2) The constraints (Assign) always hold; and
(I3) Once a constraint in (Capacity) becomes tight, it remains tight, and
(I4) Once a constraint is dropped in some iteration, it is never reinstated.

Iteration $(h+1)$ of **Sched-Cap** consists of three main steps:

1. We first remove all $X^h_{i,j} \in \{0,1\}$; i.e., we project X^h to those co-ordinates (i,j) for which $X^h_{i,j} \in (0,1)$, to obtain the current vector Y of "floating" (to-be-rounded) variables; let $\mathcal{S} \equiv (A_h Y = u_h)$ denote the current linear system that represents **LP-Cap**. (A_h is some matrix and u_h is a vector; we avoid using "$\mathcal{S}_h$" to simplify notation.) In particular, the "capacity" of machine i in $\mathcal{S}$ is its residual capacity b'_i, i.e., b_i minus the number of jobs that have been permanently assigned to i thus far. Note that the cost constraint is not included in the constraint matrix $A_h Y = u_h$, which we continue to maintain exactly. Nevertheless since all the variables maintains its initial assignment on expectation, the expected cost remains unaltered. The entire process as we demonstrate at the end can be derandomized and hence the cost upper bound of C is obeyed.

2. Let $Y \in \Re^v$ for some v; note that $Y \in (0,1)^v$. Let M_k denote the set of all machines i for which exactly k of the values $Y_{i,j}$ are positive. We will now drop some of the constraints in $\mathcal{S}$:

(D1) for each $i \in M_1$, we drop its load and capacity constraints from $\mathcal{S}$;

(D2) for each $i \in M_2$, we drop its load constraint and rewrite its capacity constraint as $x_{i,j_1} + x_{i,j_2} \leq \lceil X^h_{i,j_1} + X^h_{i,j_2} \rceil$, where j_1, j_2 are the two jobs fractionally assigned to i.

(D3) for each $i \in M_3$ for which *both* its load and capacity constraints are tight in $\mathcal{S}$, we drop its load constraint from $\mathcal{S}$.

3. Let $\mathcal{P}$ denote the polytope defined by this reduced system of constraints. A key claim that is proven in Lemma 2.3 below is that Y is *not* a vertex of $\mathcal{P}$. We now invoke **RandMove**$(Y, \mathcal{P})$; this is allowable if Y is indeed not a vertex of $\mathcal{P}$.

The above three steps complete iteration $(h+1)$.

Analysis. It is not hard to verify that the invariants (**I1**)-(**I4**) hold true (though the fact that we drop the all-important capacity constraint for machines $i \in M_1$ may look bothersome, a moment's reflection shows that such a machine cannot have a tight capacity-constraint since its sole relevant job j has value $Y_{i,j} \in (0,1)$). Since we make at least one further constraint tight via **RandMove** in each iteration, invariant (**I4**) shows that we terminate, and that the number of iterations is at most the initial number of constraints. Let us next present Lemma 2.3, a key lemma:

LEMMA 2.3. *In no iteration is Y a vertex of the current polytope $\mathcal{P}$.*

Proof. Suppose that in a particular iteration, Y **is** a vertex of $\mathcal{P}$. Fix the notation v, M_k etc. w.r.t. this iteration; let $m_k = |M_k|$, and let n' denote the remaining number of jobs that are yet to be assigned permanently to a machine. Let us lower- and upper-bound the number of variables v. On the one hand, we have $v = \sum_{k \geq 1} k \cdot m_k$, by definition of the sets M_k; since each remaining job j contributes at least two variables (co-ordinates for Y), we also have $v \geq 2n'$. Thus we get

$$v \geq n' + \sum_{k \geq 1} (k/2) \cdot m_k. \tag{2.1}$$

On the other hand, since Y has been assumed to be a vertex of $\mathcal{P}$, the number t of constraints in $\mathcal{P}$ that are satisfied *tightly* by Y, must be at least v. How large can t be? Each current job contributes one (Assign) constraint to t; by our "dropping constraints" steps (**D1**), (**D2**) and (**D3**) above, the number of tight constraints ("load" and/or "capacity") contributed by the machines is at most $m_2 + m_3 + \sum_{k \geq 4} 2m_k$.

Thus we have

$$v \leq t \leq n' + m_2 + m_3 + \sum_{k \geq 4} 2m_k. \tag{2.2}$$

Comparison of (2.1) and (2.2) and a moment's reflection shows that such a situation is possible only if: (i) $m_1 = m_3 = 0$ and $m_5 = m_6 = \cdots = 0$; (ii) the capacity constraints are tight for all machines in $M_2 \cup M_4$ – i.e., for all machines; and (iii) $t = v$. However, in such a situation, the t constraints in $\mathcal{P}$ constitute the *tight* assignment constraints for the jobs and the *tight* capacity constraints for the machines, and are hence linearly dependent (since the total assignment "emanating from" the jobs must equal the total assignment "arriving into" the machines). Thus we reach a contradiction, and hence Y is not a vertex of $\mathcal{P}$. □

We next show that the final makespan is at most $2T$ with probability one:

LEMMA 2.4. *Let X denote the final rounded vector. Algorithm **Sched-Cap** returns a schedule, where with probability one: (i) all capacity-constraints on the machines are satisfied, and (ii) for all i, $\sum_{j \in J} X_{i,j} p_{i,j} < \sum_j x^*_{i,j} p_{i,j} + max_{j \in J:\ x^*_{i,j} \in (0,1)} p_{i,j}$.*

Proof. Part (i) essentially follows from the fact that we never drop any capacity constraint; the only care to be taken is for machines i that end up in M_1 and hence have their capacity-constraint dropped. However, as argued soon after the description of the three steps of an iteration, note that such a machine cannot have a tight capacity-constraint when such a constraint was dropped; hence, even if the remaining job j got assigned finally to i, its capacity constraint cannot be violated.

Let us now prove (ii). Fix a machine i. If at all its load-constraint was dropped, it must be when i ended up in M_1, M_2 or M_3. The case of M_1 is argued as in the previous paragraph. So suppose $i \in M_\ell$ for some $\ell \in \{2, 3\}$ when its load constraint got dropped. Let us first consider the case $\ell = 2$. Let the two jobs fractionally assigned on i at that point have processing times (p_1, p_2) and fractional assignments (y_1, y_2) on i, where $0 \leq p_1, p_2 \leq T$, and $0 < y_1, y_2 < 1$. If $y_1 + y_2 \leq 1$, we know that at the end, the assignment vector X will have at most one of X_1 and X_2 being one. Simple algebra now shows that $p_1 X_1 + p_2 X_2 < p_1 y_1 + p_2 y_2 + \max\{p_1, p_2\}$ as required. If $1 < y_1 + y_2 \leq 2$, then both X_1 and X_2 can be assigned and again, $p_1 X_1 + p_2 X_2 < p_1 y_1 + p_2 y_2 + \max\{p_1, p_2\}$. For the case $\ell = 3$, we know from (**I3**) and (**D3**) that its capacity-constraint must be *tight* at some integral value u at that point, and that this capacity-constraint was preserved until the end. We must have $c = 1$ or 2 here. Let us just consider the case $c = 2$; the case of $c = 1$ is similar to the case of $\ell = 2$ with $y_1 + y_2 \leq 1$. Here again, simple algebra yields that if $0 \leq p_1, p_2, p_3 \leq T$ and $0 < y_1, y_2, y_3 < 1$ with $y_1 + y_2 + y_3 = c = 2$, then for any binary vector (X_1, X_2, X_3) of Hamming weight $c = 2$, $p_1 X_1 + p_2 X_2 + p_3 X_3 < p_1 y_1 + p_2 y_2 + p_3 y_3 + \max\{p_1, p_2, p_3\}$. □

Finally we have the following lemma.

LEMMA 2.5. *Algorithm **Sched-Cap** can be derandomized to create a schedule of cost at most C.*

Proof. Let $X^h_{i,j}$ denote the value of $x_{i,j}$ at iteration h. We know for all i, j, h, $E[X^h_{i,j}] = x^*_{i,j}$, where $x^*_{i,j}$ is solution of **LP-Cap**. Therefore, at the end, we have that the total expected cost incurred is C. The procedure can be derandomized directly by the method of conditional expectation, giving an 1-approximation to the cost. □

Lemmas 2.4 and 2.5 yield Theorem 2.2.

Proof of Theorem 2.1. We now consider the full proof of Theorem 2.1. The following integer program gives an optimal matching:

$$\sum_{i,j} c_{i,j} x_{i,j} \leq C \qquad \text{(Cost)}$$

$$\sum_{i,j} x_{i,j} \geq r_j \ \forall j \qquad \text{(Assign)}$$

$$\sum_{j} p_{i,j} x_{i,j} = f_i \ \forall i \qquad \text{(Load)}$$

$$\sum_{j} x_{i,j} \leq b_i \ \forall i \qquad \text{(Capacity)}$$

$$x_{i,j} \in \{0,1\} \ \forall i,j$$

$$x_{i,j} = 0 \quad \text{if } p_{i,j} > l_i$$

The proof of Theorem 2.1 is quite similar to Theorem 2.2. We elaborate upon the necessary modifications. First, while removing $X^h_{i,j} \in \{0,1\}$, we update the assignment requirements of the jobs as well as the capacity constraints of the machines accordingly. The dropping rules (**D1**) and (**D3**) remain the same. However, (**D2**) is modified as follows:

(**Modified D2**) *For each $i \in M_2$, we drop its load constraint and rewrite its capacity constraint. Let j_1, j_2 be the two jobs assigned to machine i with fractional assignment x_{i,j_1} and x_{i,j_2}. Then if $x_{i,j_1} + x_{i,j_2} \leq 1$, set the capacity constraint to $x_{i,j_1} + x_{i,j_2} \leq 1$. Else if $1 < x_{i,j_1} + x_{i,j_2} < 2$, set the capacity constraint to $x_{i,j_1} + x_{i,j_2} \geq 1$.*

Lemma 2.3, Lemma 2.5 remain unchanged. We have a new Lemma 2.6 corresponding to Lemma 2.4, which we prove next.

LEMMA 2.6. *Let X denote the final rounded vector. Then X satisfies with probability one: (i) all capacity-constraints on the machines are satisfied, and (ii) for all i, $\sum_j x^*_{i,j} p_{i,j} - max_{j \in J:\ x^*_{i,j} \in (0,1)} p_{i,j} < \sum_{j \in J} X_{i,j} p_{i,j} < \sum_j x^*_{i,j} p_{i,j} + max_{j \in J:\ x^*_{i,j} \in (0,1)} p_{i,j}$.*

Proof. Part (i) is similar to Part (i) of Lemma 2.5 and follows from the facts that the capacity constraints are never violated and machines in M_1 cannot have tight capacity constraints.

Let us now prove (ii). Note that in (**Modified D2**) the upper bound on capacity constraint is maintained as in (**D2**). Hence from Lemma 2.4, we get $\sum_{j \in J} X_{i,j} p_{i,j} < \sum_j x^*_{i,j} p_{i,j} + \max_{j \in J:\ x^*_{i,j} \in (0,1)} p_{i,j}$. So we only need to show the lower bound on the load. Fix a machine i. If at all its load-constraint was dropped, it must be when i ended up in $M_1 \cup M_2 \cup M_3$. In the case of M_1, at most one job fractionally assigned to it may not be assigned in the final rounded vector. So suppose $i \in M_l$ for some $l \in \{2,3\}$ when i has its load constraint dropped. Let us first consider the case of $\ell = 2$. Let the two jobs fractionally assigned to i at that point have processing times (p_1, p_2) and fractional assignments (y_1, y_2) on i, where $0 \leq p_1, p_2 \leq T$, and $0 < y_1, y_2 < 1$. If $y_1 + y_2 \leq 1$, then at the end, none of the jobs may get assigned. Simple algebra now shows that $0 > p_1 y_1 + p_2 y_2 - \max\{p_1, p_2\}$ as required. If $1 < y_1 + y_2 \leq 2$, then at least one of the two jobs X_1 and X_2 get assigned to i and again, $p_1 X_1 + p_2 X_2 > p_1 y_1 + p_2 y_2 - \max\{p_1, p_2\}$. For the case $\ell = 3$, we know from (**I3**) and (**D3**) that i's capacity-constraint must be *tight* at some integral value u at that point, and that this capacity-constraint was preserved until the end. We must have $c = 1$ or

2 in this case. Let us just consider the case $c = 2$; the case of $c = 1$ is similar to the case of $\ell = 2$ with $y_1 + y_2 \leq 1$. Here again, simple algebra yields that if $0 \leq p_1, p_2, p_3 \leq T$ and $0 < y_1, y_2, y_3 < 1$ with $y_1+y_2+y_3 = c = 2$, then for any binary vector (X_1, X_2, X_3) of Hamming weight $c = 2$, $p_1X_1+p_2X_2+p_3X_3 > p_1y_1+p_2y_2+p_3y_3-\max\{p_1, p_2, p_3\}$. □

Lemmas 2.6 and 2.5 yield Theorem 2.1.

This completes the description of this section. We have shown through our technique of rounding how a random subgraph of a bipartite graph with hard degree-constraints can be obtained that near-optimally satisfies a collection of linear constraints and respects a given cost-budget. As a special case of this, we obtained a 2 approximation algorithm for the generalized assignment problem with hard capacity-constraints on the machines.

3. Scheduling with Outliers. In this section, we consider GAP with outliers and with a hard profit constraint [31]. Formally, the problem is as follows. Let $x_{i,j}$ be the indicator variable for job j to be scheduled on machine i. Given m machines and n jobs, where job j requires processing time of $p_{i,j}$ in machine i, incurs a cost of $c_{i,j}$ if assigned to i and provides a profit of π_j if scheduled, the goal is to minimize the makespan, $T = \max_i \sum_j x_{i,j}p_{i,j}$, subject to the constraint that the total cost $\sum_{i,j} x_{i,j}c_{i,j}$ is at most C and total profit $\sum_j \pi_j \sum_i x_{i,j}$ is at least Π. The problem is motivated from improving the scheduling performance by dropping a few outliers that may be costly to schedule.

Our main contribution here is the following:

THEOREM 3.1. *For any constant $\epsilon > 0$, there is an efficient algorithm **Sched-Outlier** that returns a schedule of profit at least Π, cost at most $C(1+\epsilon)$ and makespan at most $(2+\epsilon)T$, where T is the optimal makespan among all schedules that simultaneously have cost C and profit Π.*

This is an improvement over the work of Gupta, Krishnaswamy, Kumar and Segev [31], where they constructed a schedule with makespan $3T$, profit Π and cost $C(1+\epsilon)$. In addition, our approach also accommodates *fairness*, a basic requirement in dealing with outliers, especially when problems have to be run repeatedly. We formulate fairness via stochastic programs that specify for each job j, a lower-bound r_j on the probability that it gets scheduled. We adapt our approach to honor such requirements:

THEOREM 3.2. *There is an efficient randomized algorithm that returns a schedule of profit at least Π, expected cost at most $2C$ and makespan at most $3T$ and guarantees that for each job j, it is scheduled with probability r_j, where T is the optimal expected makespan with expected cost C and expected profit Π. If the fairness guarantee on* any one *job can be relaxed, then for every fixed $\epsilon > 0$, there is an efficient algorithm to construct a schedule that has profit at least Π, expected cost at most $C(1 + 1/\epsilon)$ and makespan at most $(2+\epsilon)T$.*

We start with Theorem 3.1 and describe the algorithm **Sched-Outlier** first. Next, we prove Theorem 3.2.

Algorithm Sched-Outlier. The algorithm starts by guessing the optimal makespan T by binary search as in [36]. If $p_{i,j} > T$, then $x_{i,j}$ is set to 0. Next pick any constant $\epsilon > 0$. The running time of the algorithm depends on ϵ and is $O(n^{\frac{1}{\epsilon^c}})$, where c is some constant. We guess all assignments (i, j) where $c_{i,j} > \epsilon' C$, with $\epsilon' = \epsilon^2$. Any valid schedule can have at most $1/\epsilon'$ pairs with assignment costs higher than $\epsilon' C$; since ϵ' is a constant, this guessing can be done in time $O((mn)^{\frac{1}{\epsilon'}}) = O(n^{\frac{1}{\epsilon^2}})$. For all (i, j) with $c_{i,j} > \epsilon' C$, let $\mathcal{G}_{i,j} \in \{0, 1\}$ be a correct guessed assignment. By

enumeration, we know the optimal $\mathcal{G}_{i,j}$. For any (i,j) with $c_{i,j} > \epsilon' C$ and $c_{i,j} \notin \mathcal{G}_{i,j}$, we set $x_{i,j} = 0$. Similarly, if $c_{i,j} > \epsilon' C$ and $c_{i,j} \in \mathcal{G}_{i,j}$, then we set $x_{i,j} = 1$.

The solution to the following integer linear program then gives an optimal solution:

$$\sum_{i,j} c_{i,j} x_{i,j} \leq C \qquad \text{(Cost)}$$

$$\sum_{i} x_{i,j} = y_j \ , \forall j \qquad \text{(Assign)}$$

$$\sum_{j} p_{i,j} x_{i,j} \leq T \ , \forall i \qquad \text{(Load)}$$

$$\sum_{j} \pi_j y_j \geq \Pi \qquad \text{(Profit)}$$

$$x_{i,j} \in \{0,1\}, y_j \in \{0,1\} \ , \forall i,j$$

$$x_{i,j} = 0 \qquad \text{if } p_{i,j} > T$$

$$x_{i,j} = \mathcal{G}_{i,j} \qquad \text{if } c_{i,j} > \epsilon' C$$

We relax the constraint "$x_{i,j} \in \{0,1\}$ and $y_j \in \{0,1\}$" to "$x_{i,j} \in [0,1]$ and $y_j \in [0,1]$" to obtain the LP relaxation **LP-Out**. We solve the LP to obtain an optimal LP solution x^*, y^*; we next show how **Sched-Outlier** rounds x^*, y^* to obtain the claimed approximation. The rounding proceeds in stages as in Section 2, and as before, each variable maintains its initial assignment in x^* on expectation over the course of rounding. Hence, there is no need to explicitly consider the cost constraint. The cost constraint is dropped, yet the cost is maintained on expectation. The entire process can be derandomized efficiently. Therefore, as long as we apply our general recipe of rounding, **RandMove**, the cost is maintained exactly. Also note that if we maintain all the assign-constraints, then the profit-constraint can be dropped and is not violated. Therefore, we consider the profit constraint if and only if there are one or more assign constraints that are dropped. Also, we only need to maintain the total profit obtained from the jobs for which the assign constraints have been dropped. We now proceed to describe the rounding on each stage formally.

Note that $x^*_{i,j} \in [0,1]$ denotes the fraction of job j assigned to machine i in x^*. Initially, $\sum_i x^*_{i,j} = y^*_j$. Initialize $X = x^*$. The algorithm is composed of several iterations; the random values at the end of iteration h of the overall algorithm are denoted by X^h. (Since y_j is given by the equality $\sum_i x_{i,j}$, X^h is effectively the set of variables.) Each iteration h (except perhaps the last one) conducts a randomized update using **RandMove** on a suitable polytope constructed from a *subset* of the constraints of **LP-Out**. Therefore, for all h except perhaps the last, we have $\mathsf{E}\left[X^h_{i,j}\right] = x^*_{i,j}$. A variable $X^h_{i,j}$ is said to be *floating* if it lies in $(0,1)$, and a job is *floating* if it is not yet finally assigned. The subgraph of (J, M, E) composed of the floating edges (i,j), naturally suggests the following notation at any point of time: machines of "degree" k in an iteration are those with exactly k floating jobs assigned fractionally, and jobs of "degree" k are those assigned fractionally to exactly k machines in iteration h. Note that since we allow $y_j < 1$, there can exist singleton (i.e., degree-1) jobs which are floating.

Suppose we are at the beginning of some iteration $(h+1)$ of the overall algorithm; so we are currently looking at the values $X^h_{i,j}$. We will maintain the following invariants:

Invariants across iterations:

(I1') Once a variable $x_{i,j}$ gets assigned to 0 or 1, it is never changed;
(I2') If j is not a singleton, then $\sum_i x_{i,j}$ remains at its initial value;
(I3') The constraint (Profit) always holds;
(I4') Once a constraint is dropped, it is never reinstated.

Algorithm **Sched-Outlier** starts by initializing with **LP-Out**. Iteration $(h+1)$ consists of four major steps.

1. We remove all $X^h_{i,j} \in \{0,1\}$ as in Section 2 , i.e., we project X^h to those co-ordinates (i,j) for which $X^h_{i,j} \in (0,1)$, to obtain the current vector Z of "floating" variables; let $\mathcal{S} \equiv (A_h Z = u_h)$ denote the current linear system that represents **LP-Out**. (A_h is some matrix and u_h is a vector.)

2. Let $Z \in \Re^v$ for some v; note that $Z \in (0,1)^v$. Let M_k and N_k denote the set of degree-k machines and degree-k jobs respectively, with $m_k = |M_k|$ and $n_k = |N_k|$. We will now drop/replace some of the constraints in $\mathcal{S}$:

(D1') for each $i \in M_1$, we drop its load constraint from $\mathcal{S}$;
(D2') for each $i \in N_1$, we drop its assignment constraint from $\mathcal{S}$; we add one profit constraint (if already exists, we replace the old one) ,

$$\sum_{j \in N_1} Z_{i,j}\pi_j = \sum_{j \in N_1} X^h_{i,j}\pi_j.$$

(Note that at this point, the values $X^h_{i,j}$ are some known values.)

Thus, the assignment constraints of the singleton jobs are replaced by *one* profit constraint. As we noted earlier, it is not required to maintain the contribution to profit by the non-singleton jobs for which the assignment constraints are maintained explicitly.

3. If Z is a vertex of $\mathcal{S}$ then define the fractional assignment of a machine i by $h_i = \sum_{j \in J} Z_{i,j}$. Define a job j to be tight if $\sum_{i \in M} Z_{i,j} = 1$. Drop all the assignment constraints of the non-tight jobs (denoted by J_N) and maintain a single profit constraint,

$$\sum_{j \in N_1 \cup J_N} Z_{i,j}\pi_j = \sum_{j \in N_1 \cup J_N} X^h_{i,j}\pi_j.$$

While there exists a machine i' whose degree d satisfies $h_{i'} \geq (d - 1 - \epsilon)$, drop the load constraint on machine i'.

4. Let $\mathcal{P}$ denote the polytope defined by this reduced system of constraints. If Z is not a vertex of $\mathcal{P}$, invoke **RandMove**$(Z, \mathcal{P})$. Else we proceed differently depending on the configuration of machines and jobs in the system. If none of the following configurations is achieved (which we will show never happens at a vertex), then we report error and exit. There are five possible configurations.

- **Config-1**: *Machine and job nodes form disjoint cycles.*

Orient the edges in the bipartite graph to assign the remaining jobs in a way, so that each machine gets at most one extra job. Note that such an orientation is easy in disjoint cycles since they have even length.

- **Config-2**: *Machine and job nodes form disjoint cycles and has exactly one path with both end-points being job nodes. Thus there are two singleton jobs.*

Discard one among the two singleton jobs that has less profit. Again orient the edges in the remaining bipartite graph to assign the remaining jobs in a way, so that

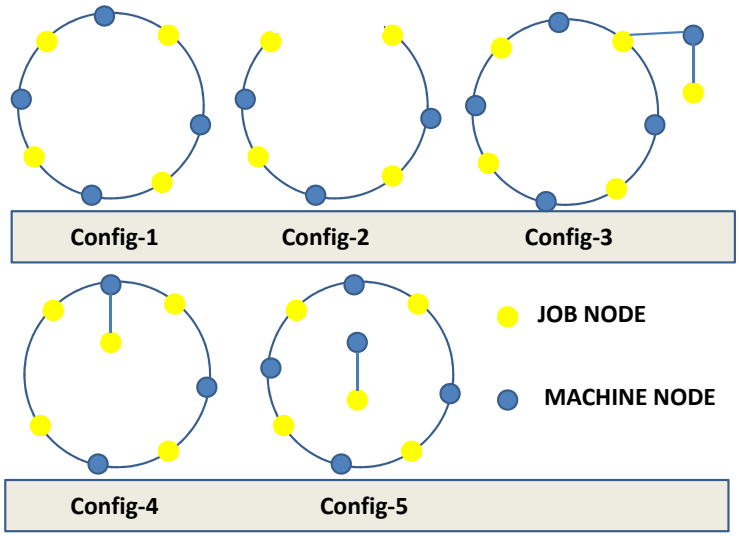

FIG. 3.1. *Different configurations of machine-job bipartite graph at step 4 of Sched-Outlier*

each machine gets at most one extra job. Such an orientation is easy in disjoint cycles (they have even length) and paths with equal number of machines and nodes.

- **Config-3**: *There is exactly one job of degree-3 and one singleton job. Rest of the jobs have degree* 2 *and all the machines have degree-2.*

Assign the singleton job to the degree-2 machine it is fractionally attached to and remove the other edge (but not the job) associated with that machine. We are left with disjoint cycles. Orient the edges in the cycles of the bipartite graph to assign the remaining jobs in a way, so that each machine gets at most one extra job.

- **Config-4**: *There is only one degree-3 machine with one singleton job. Rest of the machines have exactly two non-singleton jobs attached to it fractionally. Each non-singleton job is attached fractionally to exactly two machines.*

Assign the singleton job and the cheaper (less processing time) of the two non-singleton jobs to the degree-3 machines. Rest of the jobs and the machines form disjoints cycles in the machine-job bipartite graph or form disjoint paths each with equal number of machines and jobs in it. Orient the edges in this remaining bipartite graph in a way such that each machine gets one among the two jobs fractionally attached to it.

- **Config-5**: *Machine and job nodes form disjoint cycles. There is one extra edge with one singleton job and one singleton machine.*

Assign the singleton job to the singleton machine. Orient the edges in the cycles of the bipartite graph to assign the remaining jobs in a way, so that each machine gets at most one extra job.

The different configurations are shown pictorially in Figure 3.

Analysis. Analysis follows the following structure. First, we prove a key lemma, Lemma 3.3, which shows that if Z is a vertex and the algorithm reaches step 4, then

one of the five configurations as described above happens and also the number of machines is less than $\frac{1}{\epsilon}$. Lemma 3.3 is followed by Lemma 3.4. Lemma 3.4 establishes that the dropping and the modification of constraints in step 2 and 3, along with the assignment of jobs in step 4 do not violate the load constraint by more than a factor of $(2+\epsilon)$ and maintain the profit constraint. Lemma 3.5 bounds the cost.

Recall that in the bipartite graph $G = (J, M, E)$, we have in iteration $(h+1)$ that $(i,j) \in E$ iff $X^h_{i,j} \in (0,1)$. Any job or machine having degree 0 is thus not part of G. We prove Lemma 3.3 next.

LEMMA 3.3. *Let m denote the number of machine-nodes in G. If $m \geq \frac{1}{\epsilon}$, then Z is not a vertex of the polytope at the beginning of step 4.*

Proof. Let us consider the different possible configurations of G, when Z becomes a vertex of the polytope $\mathcal{P}$ at the beginning of step 3. There are several cases to consider depending on the number of singleton floating jobs in G in that iteration.

Case 1: There is no singleton job: We have $n_1 = 0$. Then, the number of constraints in $\mathcal{S}$ is

$$EQ = \sum_{k\geq 2} m_k + \sum_{k\geq 2} n_k.$$

Remember, since there is no singleton job, we do not consider the profit constraint explicitly. Also the number of floating variables is $v = \sum_{k\geq 2} kn_k$. Alternatively, $v = \sum_{k\geq 1} km_k$. Therefore,

$$v = \sum_{k\geq 2} \frac{k}{2}(m_k + n_k) + \frac{m_1}{2}.$$

Z being a vertex of $\mathcal{P}$, $v \leq EQ$. Thus, we must have, $n_k, m_k = 0, \quad \forall k \geq 3$ and $m_1 = 0$. Hence, every floating machine has exactly two floating jobs assigned to it and every floating job is assigned exactly to two floating machines. This is handled by Config-1.

Case 2: There are at least 3 singleton jobs: We have $n_1 \geq 3$. Then the number of linear constraints is $EQ = \sum_{k\geq 2} m_k + \sum_{k\geq 2} n_k + 1$. The last "1" comes from considering one profit constraint for the singleton jobs. The number of floating variables v again by the averaging argument as above is

$$v = \frac{n_1}{2} + \sum_{k\geq 2} \frac{k}{2}(m_k + n_k) + \frac{m_1}{2} \geq \frac{3}{2} + \sum_{k\geq 2} \frac{k}{2}(m_k + n_k) + \frac{m_1}{2}.$$

Hence, the system is always underdetermined and Z cannot be a vertex of $\mathcal{P}$.

Case 3: There are exactly 2 singleton jobs: We have $n_1 = 2$. Then the number of linear constraints is

$$EQ = \sum_{k\geq 2} m_k + \sum_{k\geq 2} n_k + 1.$$

Again the last "1" comes from considering one profit constraint for the singleton jobs. The number of floating variables v by the averaging argument is

$$v = \frac{n_1}{2} + \sum_{k\geq 2} \frac{k}{2}(m_k + n_k) + \frac{m_1}{2} \geq 1 + \sum_{k\geq 2} \frac{k}{2}(m_k + n_k) + \frac{m_1}{2}.$$

Thus, we must have, $n_k = 0, m_k = 0, \forall k \geq 3$ and $m_1 = 0$. Hence every floating machine has exactly two floating jobs assigned to it and each job except two is assigned to exactly two machines fractionally This is handled by Config-2.

Case 4: There is exactly 1 singleton job: We have $n_1 = 1$. Then the number of linear constraints is

$$EQ = \sum_{k \geq 2} m_k + \sum_{k \geq 2} n_k + 1.$$

The number of floating variables is,

$$v \geq \frac{1}{2} + n_2 + \frac{3}{2} n_3 + \frac{m_1}{2} + m_2 + \frac{3}{2} m_3 + \sum_{k \geq 4} \frac{k}{2} (m_k + n_k).$$

If Z is a vertex of $\mathcal{P}$, then $v \leq EQ$. There are only three possible configurations that might arise in this case.

(i) Only one job of degree 3 and one job of degree 1. All the other jobs have degree 2 and all the machines have degree 2. This is handled by Config-3.

(ii) Only one machine of degree 3 and one job of degree 1. The rest of the jobs and machines have degree 2. This is handled by Config-4.

(iii) Only one machine of degree 1 and one job of degree 1. The rest of the jobs and machines have degree 2. This is handled by Config-5.

Each configuration can have an arbitrary number of vertex disjoint cycles. In all these configurations, it is easy to check that for Z to be a vertex, $v = EQ$. Thus if just one constraint can be dropped, then the system becomes underdetermined.

Since we have reached a vertex at the beginning of step 3, we drop,

- All the assignment constraints for the non-tight jobs.
- Any machine i' that has degree d (where $d > 0$ is a positive integer) and the total fractional assignment from all the jobs fractionally assigned to it is at least $d - 1 - \epsilon$ loses its load-constraint.
- If the profit constraint is not already considered and some non-tight job loses its assignment constraint; we add the profit constraint.

Now we have $v = EQ$ at the beginning of step 3 and at the beginning of step 4 as well. Hence it implies either *we have not been able to drop any constraint*, or *we have dropped one assignment constraint for a non-tight job and have added one profit constraint*. We will now show that when $m \geq \frac{1}{\epsilon}$, we always drop more constraints than we add. This will give a contradiction.

In any configuration, if there is a cycle with all tight jobs, then there always exists a machine with total fractional assignment 1 and hence its load constraint can always be dropped to make the system underdetermined. So we assume there is no such cycle in any configurations. Now suppose the algorithm reaches Config-1. If there are two non-tight jobs, then we drop two assignment constraints and only add one profit constraint. Thus the system becomes underdetermined. Therefore, there can be at most one non-tight job and only one cycle (say C) with that non-tight job. Let C have m machines and thus m jobs. Therefore, $\sum_{i,j \in C} x_{i,j} \geq m - 1$. Thus there exists a machine, such that the total fractional assignment of jobs on that machine is $\geq \frac{m-1}{m} = 1 - 1/m$. If $m \geq \frac{1}{\epsilon}$, then there exists a machine with degree 2 and with total fractional assignment $\geq (1 - \epsilon)$. Thus the load-constraint on that machine gets dropped making the system underdetermined.

If the algorithm reaches Config-2, then all the non-singleton jobs must be tight for Z to be a vertex. If there are m machines, then the number of non-singleton jobs

is $m-1$. Let the two singleton jobs be j_1 and j_2. Let the two machines to which jobs j_1 and j_2 are fractionally attached with be i_1 and i_2 respectively. If $x_{i_1,j_1}+x_{i_2,j_2} \geq 1$, then the total fractional assignment from all the jobs in the system is m. Thus the machine with maximum fractional assignment must have an assignment at least 1. Since the same machine has degree 2, its load constraint gets dropped. Otherwise, the total fractional assignment from all the jobs in the system is at least $m-1$. Thus there exists a machine, such that the total fractional assignment of jobs on that machine is $\geq \frac{m-1}{m} = 1 - 1/m$. If $m \geq \frac{1}{\epsilon}$, then there exists a machine with degree 2 and with total fractional assignment $\geq (1-\epsilon)$. Thus the load- constraint on that machine gets dropped making the system underdetermined.

For Config-3 and 5, if Z is a vertex of $\mathcal{P}$, then all the jobs must be tight and using essentially the same argument, there exists a machine with fractional assignment at least $(1-\epsilon)$ if the algorithm reaches Config-3 and there exists a machine with fractional assignment 1, if the algorithm reaches Config-5.

If the algorithm reaches Config-4, then again all the jobs must be tight. If the degree-3 machine has fractional assignment at least $2-\epsilon$, then its load constraint can be dropped to make the system underdetermined. Otherwise, the total assignment to the degree-2 machines from all the jobs in the cycle is at least $m-2+\epsilon$. Therefore, there exists at least one degree-2 machine with fractional assignment at least $\frac{m-2+\epsilon}{m-1} = 1 - \frac{1-\epsilon}{m-1} \geq 1-\epsilon$, if $m \geq \frac{1}{\epsilon}$. The load-constraint on that machine can be dropped making the system underdetermined. This completes the proof of Lemma 3.3. □

We next show that the final profit is at least Π and the final makespan is at most $(2+\epsilon)T$:

LEMMA 3.4. *Let X denote the final rounded vector. Algorithm* ***Sched-Outlier*** *returns a schedule, where with probability one, (i) the profit is at least Π, (ii) for all i, $\sum_{j\in J} X_{i,j}p_{i,j} < \sum_j x^*_{i,j}p_{i,j} + (1+\epsilon)max_{j\in J:\ x^*_{i,j}\in\{0,1\}} p_{i,j}$.*

Proof. (i) This essentially follows from the fact that whenever assignment constraint on any job is dropped, its profit constraint is included in the global profit constraint of the system. At step 4 except for one configuration (Config-2), all the jobs are always assigned. Thus the profit can not decrease in those configurations. In Config-2, since we are at a vertex the total fractional assignment from the two singleton jobs is less than 1. Otherwise the system remains underdetermined from Lemma 3.3. Thus a singleton job (say j_1) is dropped, only when G has two singleton jobs j_1, j_2 fractionally assigned to i_1 and i_2 respectively, with total assignment $x_{i_1,j_1}+x_{i_2,j_2} < 1$. Since the job with the higher profit is retained, $\pi_{j_1}x_{i_1,j_1} + \pi_{j_2}x_{i_2,j_2} \leq max\{\pi_{j_1}, \pi_{j_2}\}$.

(ii) From Lemma 3.3 and **(D1')**, load constraints are dropped from machines $i \in M_1$ and might be dropped from machine $i \in M_2 \cup M_3$. For $i \in M_1$, only the remaining job j with $X^h_{i,j} > 0$, can get fully assigned to it. Hence for $i \in M_1$, its total load is bounded by $\sum_j x^*_{i,j}p_{i,j} + \max_{j\in J: x^*_{i,j}\in\{0,1\}} p_{i,j}$. For any machine $i \in M_2 \cup M_3$, if their degree d (2 or 3) is such that, its fractional assignment is at least $d-1-\epsilon$, then by simple algebra, it can be shown that for any such machine i, its total load is at most $\sum_j x^*_{i,j}p_{i,j} + (1+\epsilon)\max_{j\in J: x^*_{i,j}\in\{0,1\}} p_{i,j}$ at the end of the algorithm. For the remaining machines consider what happens at step 4. Since this is the last iteration, if we can show that the load does not increase by too much in this last iteration, we are done. Except when Config-4 is reached, any remaining machine i gets at most one extra job, and thus its total load is bounded by $\sum_j x^*_{i,j}p_{i,j} + \max_{j\in J:\ x^*_{i,j}\in\{0,1\}} p_{i,j}$. When Config-4 is reached at step 4, if the degree-3 machine has a fractional assignment at most 1 from the two jobs in the cycle, then for any value of m, there will exist a degree-2 machine whose fractional assignment is 1, giving a contradiction. Hence, let

j_1, j_2, j_3 be the three jobs assigned fractionally to the degree-3 machine i and let j_3 be the singleton job, and $x_{i,j_1} + x_{i,j_2} > 1$. If $p_{i,j_1} \leq p_{i,j_2}$, then the degree-3 machine gets j_1, j_3. Else the degree-3 machine gets j_2, j_3. The degree-3 machine gets 2 jobs, but its fractional assignment from j_1 and j_2 is already at least 1. Since the job with less processing time among j_1 and j_2 is assigned to i, its increase in load can be at most $\sum_j x^*_{i,j} p_{i,j} + \max_{j \in J:\ x^*_{i,j} \in \{0,1\}} p_{i,j}$. This completes the proof of Lemma 3.4 □

Finally we have the following lemma.

LEMMA 3.5. *Algorithm **Sched-Outlier** can be derandomized to output a schedule of cost at most $C(1+\epsilon)$.*

Proof. In all iterations h, except the last one, for all i, j, $E[X^h_{i,j}] = x^*_{i,j}$, where $x^*_{i,j}$ is solution of **LP-Out**. Therefore, before the last iteration, we have that the total expected cost incurred is C. The procedure can be derandomized directly by the method of conditional expectation, giving an 1-approximation to cost, just before the last iteration. Now at the last iteration, since at most $\frac{1}{\epsilon}$ jobs are assigned and each assignment requires at most $\epsilon' C = \epsilon^2 C$ in cost, the total increase in cost is at most ϵC, giving the required approximation. □

Lemmas 3.4 and 3.5 yield Theorem 3.1.

We now consider Theorem 3.2 that maintains fairness in the allocation of jobs while handling outliers.

Proof of Theorem 3.2

Proof. In order to maintain the scheduling probabilities of the jobs, we do not guess the assignment of jobs with high cost (cost higher than $\epsilon' C$ as in Theorem 3.1). For Part (i), we consider the first two steps of Algorithm **Sched-Outlier**. If $\mathcal{P}$ denote the polytope defined by the reduced system of constraints and the current vector Z is not a vertex of $\mathcal{P}$, then we invoke **RandMove**(Z, P) and proceed. Else from Lemma 3.3, Z is a vertex of $\mathcal{P}$ only if one of the configurations, Config-1 to Config-5, as described in step 4 of Algorithm **Sched-Outlier** is achieved and $m < \frac{1}{\epsilon}$. For any singleton job, we assign the singleton job to the corresponding machine with probability equal to its fractional assignment. Thus Theorem 3.2 remains valid for these singleton jobs. For each non-singleton job, we consider the machines to which it is fractionally assigned and allocate it to the machine which has cheaper assignment cost for it. If the algorithm reached Config-1, 2, 3 or 5, each machine can get at most two extra jobs and the expected cost is maintained. However if the algorithm reached Config-4 and the three jobs associated with the degree-3 machine were all assigned to it, then we remove one non-singleton job from the degree-3 machine. This job is assigned to the degree-2 machine in the cycle on which it had non-zero fractional assignment. This may increase the expected cost by a factor of 2 but ensures that each machine gets at most 2 additional jobs.

For Part (ii), note that the cost is maintained until the last iteration. In the last iteration, since at most $\frac{1}{\epsilon}$ jobs are assigned and each assignment requires at most C cost, we get the desired result. □

4. Max-Min Fair Allocation. In this section we consider another application of our proposed rounding method, the max-min fair allocation problem [14, 15, 5, 4, 8]. In this problem there are m goods that need to be distributed indivisibly among k persons. Each person i has a non-negative integer valuation $u_{i,j}$ for good j. The valuation functions are linear, i.e., $u_{i,C} = \sum_{j \in C} u_{i,j}$ for any set of C goods. The goal is to allocate each good to a person such that the "least happy person is as happy as possible": i.e., $\min_i u_{i,C}$ is maximized. Our main contribution in this regard is to near-optimally pin-point the integrality gap of a *configuration LP* previously proposed

and analyzed in [8, 5]. Our algorithm uses bipartite dependent rounding [28] and its generalization to weighted graphs. Bipartite dependent rounding can be viewed as a specific type of **RandMove** on bipartite graphs. A crucial ingredient of our analysis is to show certain random variables satisfy the property of *negative correlation* and hence the Chernoff type concentration bounds can be applied to guarantee small deviation from the expected value of their sum.

Configuration LP for Max-Min Fair Allocation. The configuration LP formulation for the max-min fair allocation problem was first considered in [8]. A *configuration* is a subset of items and in the LP there is a variable for each valid configuration. Using binary search, first the optimum solution value T is guessed and then we define valid configurations based on the approximation factor λ sought for. We call a configuration C *valid* for person i if either of the following two conditions hold:

- $u_{i,C} \geq T$ and all the items in C have value at most $\frac{T}{\lambda}$. These are called *small* items.
- C contains only one item j and $u_{i,j} \geq \frac{T}{\lambda}$. We call such an item j to be a *big* item for person i.

We define a variable $x_{i,C}$ for assigning a valid configuration C to person i. Let $C(i,T)$ denote the set of all valid configurations corresponding to person i with respect to T. The configuration LP relaxation of the problem is as follows:

$$\begin{aligned} \forall j : \sum_{C \ni j} \sum_{i} x_{i,C} &\leq 1 \qquad (4.1) \\ \forall i : \sum_{C \in C(i,T)} x_{i,C} &= 1 \\ \forall i, C : x_{i,C} &\geq 0 \end{aligned}$$

The above LP formulation may have exponential number of variables, yet if the LP is feasible, then a fractional allocation where each person receives either a big item or at least a utility of $T(1-\epsilon)$ can be computed in polynomial time [8]. Here ϵ is any constant greater than zero. In the subsequent discussion and analysis, we ignore the multiplicative $1+\epsilon$ factor; it is hidden in the Θ notation of the ultimate approximation ratio.

The integrality gap of the above configuration LP is $\Omega(\frac{1}{\sqrt{k}})$ and again follows from [8]. In [5], Asadpour and Saberi gave a rounding procedure for the configuration LP that achieved an approximation factor of $O\left(\frac{1}{\sqrt{k}(\ln k)^3}\right)$. Here we further lower the gap and prove the following theorem.

THEOREM 4.1. *Given any feasible solution to the configuration LP, it can be rounded to a feasible integer solution such that every person gets at least* $\Theta\left(\frac{1}{\sqrt{k \ln k}}\right)$ *fraction of the optimal utility with probability at least* $1 - \Theta(\frac{1}{k})$ *in polynomial time.*

Our proof is also significantly simpler than the one in [5].

Note that the recent work of Chakrabarty, Chuzhoy and Khanna [20] has an improved approximation factor of m^{ϵ} (also note that $m \geq k$) but that does not use the configuration LP.

In the context of fair allocation, an additional important criterion can be an *equitable partitioning* of goods: we may impose an upper bound on the number of items a person might receive. For example, we may want each person to receive at

most $\lceil \frac{m}{k} \rceil$ goods. Theorem 2.1 leads to the following:

THEOREM 4.2. *Suppose, in max-min allocation, we are given upper bounds c_i on the number of items that each person i can receive, in addition to the utility values $u_{i,j}$. Let T be the optimal max-min allocation value that satisfies c_i for all i. Then, we can efficiently construct an allocation in which for each person i the bound c_i holds and he receives a total utility of at least $T - \max_j u_{i,j}$.*

This generalizes the result of [14], which yields the "$T - \max_j u_{i,j}$" value when no bounds such as the c_i are given. To our knowledge, the results of [15, 5, 4, 8] do not carry over to the setting of such "fairness bounds" c_i.

4.1. Algorithm for Max-Min Fair Allocation. We now describe the algorithm and the proof of Theorem 4.1.

4.1.1. Algorithm. We define a weighted bipartite graph G with the vertex set $A \bigcup B$ corresponding to the persons and the items respectively. There is an edge between a vertex corresponding to person $i \in A$ and item $j \in B$, if a configuration C containing j is fractionally assigned to i. Define

$$w_{i,j} = \sum_{C \ni j} x_{i,C},$$

i.e., $w_{i,j}$ is the fraction of item j that is allocated to person i by the fractional solution of the LP. An edge (i, j) is called a matching edge, if the item j is big for person i. Otherwise it is called a flow edge.

Let M and F represent the set of matching and flow edges respectively. For each vertex $v \in A \bigcup B$, let m_v be the total fraction of the matching edges incident to it. Also define $f_v = 1 - m_v$. The main steps of the algorithm are as follows,

1. Guess the value of the optimum solution T by doing a binary search. Solve LP (4.1). Obtain the set M and m_v, f_v for each vertex v in G constructed from the LP (4.1) solution.

2 **Allocating Big Items** : Select a random matching from edges in M using *bipartite dependent rounding* (Section 4.1.2) such that for every $v \in A \bigcup B$, the probability that v is saturated by the matching is $m_v = 1 - f_v$.

3 **Allocating Small Items** :

   (a) Discard any item j, with $m_j \geq (1 - \epsilon_1)$, $\epsilon_1 = \sqrt{\frac{\ln k}{k}}$, and also discard all the persons and the items saturated by the matching.
   (b) (Scaling) In the remaining graph containing only flow edges for unsaturated persons and items, set for each person i, $w'_{i,j} = \frac{w_{i,j}}{f_i}$, $\forall j$.
   (c) Further discard any item j with $\sum_i w'_{i,j} \geq \frac{3 \ln k}{\ln \ln k}$.
   (d) (weighted bipartite dependent rounding) Scale down the weights on all the remaining edges by a factor of $\frac{3 \ln k}{\ln \ln k}$ and do a *weighted bipartite dependent rounding* to assign items to persons.

We now analyze each step. The main proof idea is in showing that there remains enough left-over utility in the flow graph for each person not saturated by the matching. This is obtained through proving a new negative correlation property among the random variables defined on a collection of vertices. Previously, the negative correlation property due to bipartite dependent rounding has only been proven for variables

defined on edges incident on any particular vertex. Such "local" negative correlation property is not sufficient for our case.

4.1.2. Allocating Big Items. Consider the edges in M in the person-item bipartite graph. Remove all the edges (i, j) that have already been rounded to 0 or 1. Additionally, if an edge is rounded to 1, remove both its endpoints i and j. We initialize for each $(i, j) \in M$, $y_{i,j} = w_{i,j}$, and modify the $y_{i,j}$ values probabilistically in rounds using bipartite dependent rounding.

Bipartite Dependent Rounding. The bipartite dependent rounding selects an even cycle $\mathcal{C}$ or a maximal path $\mathcal{P}$ in G, and partitions the edges in $\mathcal{C}$ or $\mathcal{P}$ into two matchings $\mathcal{M}_1$ and $\mathcal{M}_2$. Then, two positive scalars α and β are chosen as follows:

$$\alpha = \min\{\eta > 0 : ((\exists (i,j) \in \mathcal{M}_1 : y_{i,j} + \eta = 1) \bigcup (\exists (i,j) \in \mathcal{M}_2 : y_{i,j} - \eta = 0))\};$$

$$\beta = \min\{\eta > 0 : ((\exists (i,j) \in \mathcal{M}_1 : y_{i,j} - \eta = 0) \bigcup (\exists (i,j) \in \mathcal{M}_2 : y_{i,j} + \eta = 1))\};$$

Now with probability $\frac{\beta}{\alpha+\beta}$, set

$$\begin{aligned} y'_{i,j} &= y_{i,j} + \alpha \text{ for all } (i,j) \in \mathcal{M}_1 \\ \text{and } y'_{i,j} &= y_{i,j} - \alpha \text{ for all } (i,j) \in \mathcal{M}_2; \end{aligned}$$

with complementary probability of $\frac{\alpha}{\alpha+\beta}$, set

$$\begin{aligned} y'_{i,j} &= y_{i,j} - \beta \text{ for all } (i,j) \in \mathcal{M}_1 \\ \text{and } y'_{i,j} &= y_{i,j} + \beta \text{ for all } (i,j) \in \mathcal{M}_2; \end{aligned}$$

The above rounding scheme satisfies the following two properties, which are easy to verify:

$$\forall\, i, j,\ \mathsf{E}\big[y'_{i,j}\big] = y_{i,j} \tag{4.2}$$

$$\exists\, i, j,\ y'_{i,j} \in \{0, 1\} \tag{4.3}$$

Thus, if $Y_{i,j}$ denotes the final rounded values then Property (4.2) guarantees for every edge (i, j), $\mathsf{E}\big[Y_{i,j}\big] = w_{i,j}$. This gives the following corollary.

COROLLARY 4.3. *The probability that a vertex $v \in A \bigcup B$ is saturated in the matching generated by the algorithm is m_v.*

Proof. Let there be $l \geq 0$ edges $e_1, e_2, ..e_l \in M$ that are incident on v. Then,

$$\begin{aligned} \mathsf{Pr}\big[v \text{ is saturated}\big] &= \mathsf{Pr}\big[\exists\, e_i, i \in [1,l] \ \ s.t \ \ v \text{ is matched with } \ e_i\big] \\ &= \sum_{i=1}^{l} \mathsf{Pr}\big[v \text{ is matched with } e_i\big] = \sum_{i=1}^{l} w_i = m_v \end{aligned}$$

Here the second equality follows by replacing the union bound by sum since the events are mutually exclusive. □

Now we prove two additional properties of this rounding, which will be used crucially for the analysis of the next step.

DEFINITION 4.4 (Negative Correlation for Indicator Random Variables). *A collection of indicator random variables $\{z_i\}, i \in [1,n]$ are said to be negatively correlated, if for any subset of t variables, $t \in [1,n]$, and any $b \in \{0,1\}$, $\mathsf{Pr}\big[\bigwedge_{j=1}^{t} z_{i_j} = b\big] \leq \prod_{j=1}^{t} \mathsf{Pr}\big[z_{i_j} = b\big]$.*

THEOREM 4.5. *Define an indicator random variable z_j for each item $j \in B$ with $m_j < 1$, such that $z_j = 1$ if item j is saturated by the matching. Then, the indicator random variables $\{z_j\}$ are negatively correlated.*

Proof. Consider any collection of items $j_1, j_2, \ldots, j_t$. Let $b = 1$ (the proof for the case $b = 0$ is identical). Let $y_{i,j,k}$ denote the value of $y_{i,j}$ at the beginning of the k-th iteration of bipartite dependent rounding. Define, $z_{j,k} = \sum_{i,(i,j)\in M} y_{i,j,k}$. Clearly, $z_j = \sum_{i,(i,j)\in M} y_{i,j,|M|+1}$. We will show that

$$\forall k, \mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,k}\big] \leq \mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,k-1}\big] \tag{4.4}$$

Thus, we will have

$$\begin{aligned}
\mathsf{Pr}\big[\bigwedge_{i=1}^{t} z_{j_i} = 1\big] &= \mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,|M|+1}\big] \\
&\leq \mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,1}\big] \\
&= \prod_{i=1}^{t} \sum_{v} y_{v,j_i,1} = \prod_{i=1}^{t} m_{j_i} = \prod_{i=1}^{t} \mathsf{Pr}\big[z_{j_i} = 1\big]
\end{aligned}$$

We now prove (4.4) for a fixed k. Note that any vertex that is not the end point of the maximal path or the cycle on which dependent rounding is applied on the $k-1$-th round retains their previous z value. There are three cases to consider:

Case 1: *Two vertices among $j_1, j_2, \ldots, j_t$ have their values modified.* Let these vertices be say j_1 and j_2. Therefore, these two vertices must be the end points of the maximal path on which dependent rounding is applied on the $k-1$-th round. The path length must be even. Let $B(j_1, j_2, \alpha, \beta)$ denote the event that the jobs $\{j_1, j_2\}$ have their values modified in the following probabilistic way:

$$(z_{j_1,k}, z_{j_2,k}) = \begin{cases} (z_{j_1,k-1} + \alpha, z_{j_2,k-1} - \alpha) & \text{with probability } \frac{\beta}{\alpha+\beta} \\ (z_{j_1,k-1} - \beta, z_{j_2,k-1} + \beta) & \text{with probability } \frac{\alpha}{\alpha+\beta} \end{cases}$$

Thus

$$\begin{aligned}
&\mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,k} | \forall i \in [1,t],\ z_{j_i,k-1} = a_{j_i} \wedge B(j_1, j_2, \alpha, \beta)\big] \\
&= \mathsf{E}\big[z_{j_1,k} z_{j_2,k} | \forall i \in [1,t],\ z_{j_i,k-1} = a_{j_i} \wedge B(j_1, j_2, \alpha, \beta)\big] \prod_{i=3}^{t} a_{j_i}
\end{aligned}$$

The above expectation can be written as $(\psi + \phi)\Pi_{i=3}^{t} a_{j_i}$, where

$$\psi = (\beta/(\alpha+\beta))(a_{f_1} + \alpha)(a_{f_2} - \alpha), \text{ and}$$

$$\phi = (\alpha/(\alpha+\beta))(a_{f_1}+\beta)(a_{f_2}+\beta).$$

Now, it can be easily seen that $\psi + \phi \leq a_{j_1} a_{j_2}$. Thus for any fixed j_1, j_2 and for any fixed (α, β), and for fixed values a_f the following holds:

$$\mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,k} | \forall i \in [1,t],\ z_{j_i,k-1} = a_j \wedge B(j_1, j_2, \alpha, \beta)\big] \leq \prod_{i=1}^{t} a_j.$$

Hence, $\mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,k} | \text{Case 1}\big] \leq \mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,k-1} | \text{Case 1}\big]$.

Case 2: *One vertex among $j_1, j_2, \ldots, j_t$ has its value modified.* Let the vertex be j_1 say. Therefore, this vertex must be the end point of the maximal path on which dependent rounding is applied on the $(k-1)$-th round. The path length must be odd. Let $B(j_1, \alpha, \beta)$ denote the event that the job j_1 has its value modified in the following probabilistic way:

$$z_{j_1,k} = \begin{cases} z_{j_1,k-1} + \alpha & \text{with probability } \frac{\beta}{\alpha+\beta} \\ z_{j_1,k-1} - \beta & \text{with probability } \frac{\alpha}{\alpha+\beta} \end{cases}$$

Thus,

$$\mathsf{E}\big[z_{j_1,k} | \forall i \in [1,t],\ z_{j_i,k-1} = a_j \wedge B(j_1, \alpha, \beta)\big] = a_{j_1}.$$

Since the values of z_{j_i}, $i \in [2,t]$ remains unchanged and the above equation holds for any j_1, α, β, we have $\mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,k} | \text{Case 2}\big] \leq \mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,k-1} | \text{Case 2}\big]$.

Case 3: *None among $j_1, j_2, \ldots, j_t$ has its value modified.*

In this case, the value of $z_{j_i,k}$'s, $i \in [1,t]$, do not change. Hence, $\mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,k} | \text{Case 3}\big] \leq \mathsf{E}\big[\prod_{i=1}^{t} z_{j_i,k-1)}\big]$.

This establishes the claim.

□

As a corollary of the above theorem, Theorem 4.5, we get the following claim.

COROLLARY 4.6. *Define an indicator random variable z_i for each person $i \in A$, such that $z_i = 1$ if person i is saturated by the matching. Then, the indicator random variables $\{z_i\}$ are negatively correlated.*

Proof. Do the same analysis as in Theorem 4.5 with items replaced by persons. □

4.1.3. Allocating small items. We now prove in Lemma 4.7 that after the matching phase, each unsaturated person has available items with utility at least $\sqrt{\frac{\ln k}{k}}\frac{T}{5}$ in the flow graph. Additionally we prove in Lemma 4.8 that each item is not claimed more than $3\ln k/\ln\ln k$). Both the results are probabilistic and hold with high probability. We use the following form of the well-known Chernoff-Hoeffding Bound.

The Chernoff-Hoeffding Bound[39]: *Suppose $X = \sum_i X_i$ where X_i are independent/negatively correlated random variables taking values in $[0,1]$ with $\mathsf{E}\big[X\big] = \mu$, then for $\delta > 0$ we have*

1. $\mathsf{Pr}\big[X \geq \mu(1+\delta)\big] \leq e^{-\mu\delta^2/3}$
2. $\mathsf{Pr}\big[X \leq \mu(1-\delta)\big] \leq e^{-\mu\delta^2/2}$
3. *For $\delta \geq 1$,* $\mathsf{Pr}\big[X \geq \mu(1+\delta)\big] \leq e^{-\mu(\delta+1)\ln\delta + 1[1 - \frac{\delta}{(1+\delta)\ln(1+\delta)}]}$.

LEMMA 4.7. *After* **Allocation of Big Items** *by bipartite dependent rounding, each unsatisfied person has a total utility of at least* $\sqrt{\frac{\ln k}{k}}\frac{T}{5}$ *from the unsaturated items with probability at least* $1 - \frac{1}{k}$.

Proof. Consider a person v who is not saturated by the matching. In step (a) of **Allocation of Small Items**, all items j with m_j at least $(1-\epsilon_1)$ are discarded. We will set $\epsilon_1 = \sqrt{\frac{\ln k}{k}}$ later. Since the total sum of m_j can be at most k (the number of persons), there can be at most $\frac{k}{1-\epsilon_1}$ items with m_j at least $1-\epsilon_1$. Therefore, for the remaining items, we have $f_j \geq \epsilon_1$. Each person is connected only to small items in the flow graph. After removing the items with m_j at least $1-\epsilon_1$, the remaining utility in the flow graph for person v is at least

$$\left(T - \sum_{j: f_j \leq \epsilon_1 \text{ and } j \text{ is unsaturated}} u_{v,j} f_j\right) \geq \left(T - \frac{\epsilon_1 k}{1-\epsilon_1}\frac{T}{\lambda}\right). \tag{4.5}$$

Define $w'_{v,j} = \frac{w_{v,j}}{f_v}$ and select a $\lambda_1 \leq \lambda$. Now consider random variables $Y_{v,j}$ for each of these unsaturated items:

$$Y_{v,j} = \begin{cases} \frac{w'_{v,j} u_{v,j}}{T/\lambda_1} & : \text{if item } j \text{ is not saturated} \\ 0 & : \text{ otherwise} \end{cases} \tag{4.6}$$

Since each $u_{v,j} \leq T/\lambda \leq T/\lambda_1$ and $w_{v,j} \leq f_v$, $Y_{v,j}$ are random variables bounded in $[0,1]$. Person v is not saturated by the matching with probability $1 - m_v = f_v$. Each such person v gets a fractional utility of $w'_{v,j}u_{v,j}$ from the small (with respect to the person) item j in the flow graph, if item j is not saturated by the matching. The later happens with probability f_j.

Define $G_v = \sum_j Y_{v,j}$. Then $\frac{T}{\lambda_1}G_v$ is the total fractional utility after step (b). It follows from Equation 4.5

$$\mathsf{E}\big[G_v\big] = \sum_j \frac{w'_{v,j}u_{v,j}f_j}{T/\lambda_1} \geq \epsilon_1\lambda_1\left(1 - \frac{\epsilon_1 k}{(1-\epsilon_1)\lambda}\right)$$

Set $\epsilon_1 = \sqrt{\frac{\ln k}{k}}$, $\lambda_1 = 25\sqrt{k \ln k}$ and we have $\lambda \geq \lambda_1$.

Thus for sufficiently large k,

$$\begin{aligned} \mathsf{E}\big[G_v\big] &\geq \epsilon_1\lambda_1\left(1 - \frac{\epsilon_1 k}{(1-\epsilon_1)\lambda_1}\right) \\ &\geq 24 \ln k \end{aligned}$$

That $Y_{v,j}$'s are negatively correlated follows from Theorem 4.5. Therefore, applying the standard Chernoff-Hoeffding bound for the negatively-correlated random variables, we get for any $\delta \in (0,1)$

$$\begin{aligned} \mathsf{Pr}\big[G_v \leq (1-\delta)\mathsf{E}\big[G_v\big]\big] &\leq e^{-E[G_v]\delta^2/3} \\ &\leq e^{-24\ln k/12} \text{ for } \delta \geq \frac{1}{2} \\ &= \frac{1}{k^2}. \end{aligned}$$

Thus we get

$$\mathsf{Pr}\big[\frac{T}{\lambda_1}G_v \le \frac{1}{2}\frac{T}{\lambda_1}\mathsf{E}\big[G_v\big]\big] \le \frac{1}{k^2}$$

Hence,

$$\exists v,\ \ \mathsf{Pr}\big[\frac{T}{\lambda_1}G_v \le \frac{1}{2}\frac{T}{\lambda_1}\mathsf{E}\big[G_v\big]\big] \le \frac{1}{k}$$

Therefore the net fractional utility that remains for each person in the flow graph after scaling is at least $\frac{1}{2}\frac{T}{\lambda_1}\mathsf{E}\big[G_v\big] = \frac{1}{2}\frac{T}{25\sqrt{k\ln k}}12\ln k \ge \frac{T}{5}\sqrt{\frac{\ln k}{k}}$, with probability at least $1-\frac{1}{k}$. □

LEMMA 4.8. *After the matching and the scaling (step (b)), each unsaturated item has a total fractional incident edge-weight to be at most* $\frac{3\ln k}{\ln\ln k}$ *from the unsaturated persons with probability at least* $1-\frac{1}{k^3}$.

Proof. Note that for any person v and for any job j that is small for v, $w_{v,j} \le f_v$, hence $w'_{v,j} = \frac{w_{v,j}}{f_v} \le 1$. Fix an item j, and define a random variable $Z_{v,j}$ for each person such that

$$Z_{v,j} = \begin{cases} w'_{v,j} & : \text{if person } i \text{ is not saturated} \\ 0 & : \text{ otherwise} \end{cases} \tag{4.7}$$

Let $X_j = \sum_v Z_{v,j}$. Then X_j is the total weight of all the edges incident on item j in the flow graph after scaling and removal of all saturated persons. We have $\mathsf{E}\big[X_j\big] = \sum_v w'_{v,j}f_v = \sum_v w_{v,j} \le 1$. Now that the variables $Z_{v,j}$ are negatively correlated follows from Corollary 4.6, and thus applying the Chernoff-Hoeffding bound for the negatively correlated variables we get

$$\mathsf{Pr}\big[X_j \ge \frac{3\ln k}{\ln\ln k}\big] \le \frac{1}{k^3}$$

This completes the proof. □

Recall the third step, step (c), of **Allocating Small Bundles**. Any job in the remaining flow graph with total weight of incident edges more than $\frac{3\ln k}{\ln\ln k}$ is discarded in this step. We now calculate the utility that remains for each person in the flow graph after step (c).

LEMMA 4.9. *After removing all the items that have total degree more than* $\frac{3\ln k}{\ln\ln k}$ *in the flow graph, that is after step (c) of* **Allocating Small Items**, *the remaining utility of each unsaturated person in the flow graph is at least* $\sqrt{\frac{\ln k}{k}}\frac{T}{6}$ *with probability at least* $1-\frac{2}{k}$.

Proof. Fix a person v and consider the utility that v obtains from the fractional assignments in the flow graph before step (c). It is at least $\sqrt{\frac{1}{klnk}}\frac{T}{4}$ from Lemma 4.7. Define a random variable for each item that v claims with nonzero value in the flow graph at step (b):

$$Z'_{v,j} = \begin{cases} u_{v,j} & : \text{if item } j \text{ has total weighted degree at least } \frac{3\ln k}{\ln\ln k} \\ 0 & : \text{ otherwise} \end{cases} \tag{4.8}$$

We have $\mathsf{Pr}\big[Z'_{v,j} = u_{v,j}\big] \leq \frac{1}{k^3}$ from Lemma 4.8. Therefore, the expected utility from all the items in the flow graph that have total incident weight more than $\frac{3\ln k}{\ln\ln k}$ is at most $\frac{T}{k^3}$. Hence by Markov's inequality, the utility from the discarded items is more than $\frac{T}{k}$ is at most $\frac{1}{k^2}$. Now, by union bound, the utility from the discarded items is more than $\frac{T}{k}$ for at least one unsaturated person is at most $\frac{1}{k}$. The initial utility before step (c) was at least $\sqrt{\frac{\ln k}{k}}\frac{T}{5}$ with probability $1-\frac{1}{k}$. Thus after step (c), the remaining utility is at least $\sqrt{\frac{\ln k}{k}}\frac{T}{5} - \frac{T}{k}$ with probability at least $1-\frac{2}{k}$. Hence, the result follows. □

The next and the final step (d) of allocations is to do a weighted dependent rounding on a *scaled down* flow graph. The weight on the remaining edges is scaled down by a factor of $\frac{3\ln k}{\ln\ln k}$ and hence for every item node that survives step (c), the total edge-weight incident on it is at most *one*. Let us denote by $W_{i,j}$ the fractional weight on the edge (i,j) in this graph. Hence after scaling down the utility of any person v in the flow graph is $\sum_j u_{v,j}W_{v,j} \geq \frac{\ln\ln k}{\ln k}\sqrt{\frac{\ln k}{k}}\frac{T}{6} = \frac{\ln\ln k}{\sqrt{k\ln k}}\frac{T}{6}$.

Weighted Dependent Rounding. We remove all (i,j) that have already been rounded to 0 or 1. Let F' be the current graph consisting of those $W_{i,j}$ that lie in $(0,1)$. Choose any maximal path $P = (v_0, v_1, .., v_s)$ or a cycle $C = (v_0, v_1, .., v_s = v_0)$. The current W value of an edge $e_t = (v_{t-1}, v_t)$ is denoted by y_t, that is $y_t = W_{t-1,t}$.

We next choose the values $z_1, z_2, .., z_s$ such that any unsaturated person retains the utility it started with after scaling down, as long as there are at least two edges incident to it. We update the W value of each edge $e_t = (v_{t-1}, v_t)$ to $y_t + z_t$.

Suppose we have initialized some value for z_1 and that we have chosen the increments $z_1, z_2, \ldots, z_t$ for some $t \geq 1$. Then the value z_{t+1} corresponding to the edge $e_{t+1} = (v_t, v_{t+1})$ is chosen as follows:

- (**PI**) v_t is an item, then $v_{t+1} = -v_t$. (Each item is not assigned more than once.)
- (**PII**) v_t is a person. Then choose z_{t+1} so that the utility of w_t remains unchanged. Set $z_{t+1} = z_t\frac{-u_{v_t,v_{t-1}}}{u_{v_t,v_{t+1}}}$.

The vector $z = (z_1, z_2, ..z_s)$ is completely determined by z_1. We denote this by $f(z)$.

Now let μ be the smallest positive value such that if we set $z_1 = \mu$, then all the W values (after incrementing by the vector z as specified above) stay in $[0,1]$, and at least one of them becomes 0 or 1. Similarly, let γ be the smallest value such that if we set $z_1 = -\gamma$, then this rounding progress property holds.

When considering a cycle, assume v_0 is a person. The assignment of z_i values ensure all the objects in the cycle are assigned exactly once and utility of all the persons except v_0 remains unaffected. Now the change in the value of z_s is

$$-z_1\frac{u_{v_2,v_1}u_{v_4,v_3}...u_{v_{s-1},v_{s-2}}}{u_{v_2,v_3}u_{v_4,v_5}...u_{v_{s-1},v_s}}.$$

If

$$\frac{u_{v_2,v_1}u_{v_4,v_3}...u_{v_{s-1},v_{s-2}}}{u_{v_2,v_3}u_{v_4,v_5}...u_{v_{s-1},v_s}} > 1,$$

we set $z_1 = -\gamma$, else we set $z_1 = \mu$. Therefore the utility of the person v_0 can only increase.

When we are considering a maximal path we choose the vector z as either $f(\mu)$ or $f(-\gamma)$ arbitrarily.

LEMMA 4.10. *Each person unsaturated by the matching receives a utility of at least* $\Theta(\frac{\ln \ln k}{\sqrt{k \ln k}} T)$ *after step (d).*

Proof. While the weighted dependent rounding scheme is applied on a cycle, all persons in the remaining graph maintains their utility. Only when the rounding is applied on a maximal path, the two persons at two ends might lose one item.

Hence, the net utility received by any person after step (d) is at most $\frac{T}{\lambda}$ less than what it was just before starting the weighted dependent rounding step. Thus each person receives a utility of $\frac{\ln \ln k}{\sqrt{k \ln k}} \frac{T}{6} - \frac{T}{\lambda}$. From Lemma 4.7, $\lambda \geq 25\sqrt{k \ln k}$. Substituting $\lambda = 25\sqrt{k \ln k}$, we get the desired result. □

We have thus established Theorem 4.1. The approximation ratio is $\Theta(\frac{1}{\sqrt{k \ln k}})$. This provides an upper bound of $\sqrt{k \ln k}$ on the integrality gap of the configuration LP for the max-min fair allocation problem. In contrast, the lower bound is $\Omega(\sqrt{k})$ [8]. Theorem 4.2 that incorporates fairness in allocation by providing a limit on the cardinality of items each person can receive is a direct corollary of Theorem 2.1. Such fairness results in the context of the max-min fair allocation problem was not know earlier.

5. Designing Overlay Multicast Networks For Streaming. The work of [2] studies approximation algorithms for designing a multicast overlay network. We first describe the problem and state the results in [2] (Lemma 5.1 and Lemma 5.2). Next, we show our main improvement in Lemma 5.3.

The background text here is largely borrowed from [2]. An overlay network can be represented as a tripartite digraph $N = (V, E)$. The nodes V are partitioned into sets of entry points called sources (S), reflectors (R), and edge-servers or sinks (D). There are multiple commodities or streams, that must be routed from sources, via reflectors, to the sinks that are designated to serve that stream to end-users. Without loss of generality, we can assume that each source holds a single stream. Now given a set of streams and their respective edge-server destinations, a cheapest possible overlay network must be constructed subject to certain *capacity*, *quality*, and *reliability* requirements. There is a cost associated with usage of every link and reflector. There are capacity constraints, especially on the reflectors, that dictate the maximum total bandwidth (in bits/sec) that the reflector is allowed to send. The quality of a stream is directely related to whether or not an edge-server is able to reconstruct the stream without significant loss of accuracy. Therefore even though there is some loss threshold associated with each stream, at each edge-server only a maximum possible reconstruction-loss is allowed. To ensure reliability, multiple copies of each stream may be sent to the designated edge-servers.

All these requirements can be captured by an integer program. Let us use indicator variable z_i for building reflector i, $y_{i,k}$ for delivery of k-th stream to the i-th reflector and $x_{i,j,k}$ for delivering k-th stream to the j-th sink through the i-th reflector. F_i denotes the fanout constraint for each reflector $i \in R$. Let $p_{x,y}$ denote the failure probability on any edge (source-reflector or reflector-sink). We transform the probabilities into weights: $w_{i,j,k} = -\log(p_{k,i} + p_{i,j} - p_{k,i}p_{i,j})$. Therefore, $w_{i,j,k}$ is the negative log of the probability of a commodity k failing to reach sink j via reflector i. On the other hand, if $\phi_{j,k}$ is the minimum required success probability for commodity k to reach sink j, we instead use $W_{j,k} = -\log(1 - \phi_{j,k})$. Thus $W_{j,k}$ denotes the negative log of maximum allowed failure. r_i is the cost for opening the

$$\min \sum_{i \in R} r_i z_i + \sum_{i \in R} \sum_{k \in S} c_{k,i,k} y_{i,k} + \sum_{i \in R} \sum_{k \in S} \sum_{j \in D} c_{i,j,k} x_{i,j,k}$$

$$s.t. \tag{5.1}$$

$$y_{k,i} \leq z_i \quad \forall i \in R, \quad \forall k \in S \tag{5.2}$$

$$x_{i,j,k} \leq y_{i,k} \quad \forall i \in R, \quad \forall j \in D, \quad \forall k \in S \tag{5.3}$$

$$\sum_{k \in S} \sum_{j \in D} x_{i,j,k} \leq F_i z_i \quad \forall i \in R \tag{5.4}$$

$$\sum_{i \in R} x_{i,j,k} w_{i,j,k} \geq W_{j,k} \quad \forall j \in D, \forall k \in S \tag{5.5}$$

$$x_{i,j,k} \in \{0,1\}, y_{i,k} \in \{0,1\}, z_i \in \{0,1\} \tag{5.6}$$

TABLE 5.1
Integer Program for Overlay Multicast Network Design

reflector i and $c_{x,y,k}$ is the cost for using the link (x, y) to send commodity k. Thus we have the IP (see Table 5.1).

Constraints (5.2) and (5.3) are natural consistency requirements; constraint (5.4) encodes the fanout restriction. Constraint (5.5), the *weight* constraint, ensures quality and reliability. Constraint (5.6) is the standard integrality-constraint that will be relaxed to construct the LP relaxation.

There is an important stability requirement that is referred as *color constraint* in [2]. Reflectors are grouped into m color classes, $R = R_1 \cup R_2 \cup \ldots \cup R_m$. We want each group of reflectors to deliver not more than one copy of a stream into a sink. This constraint translates to

$$\sum_{i \in R_l} x_{i,j,k} \leq 1 \ \forall j \in D, \ \forall k \in S, \ \forall l \in [m] \tag{5.7}$$

Each group of reflectors can be thought to belong to the same ISP. Thus we want to make sure that a client is served only with one – the best – stream possible from a certain ISP. This diversifies the stream distribution over different ISPs and provides stability. If an ISP goes down, still most of the sinks will be served. We refer the LP-relaxation of integer program (Table 5.1) with the color constraint (5.7) as **LP-Color**.

All of the above is from [2].

The work of [2] uses a two-step rounding procedure and obtains the following guarantee.

First stage rounding: Rounds z_i and $y_{i,k}$ for all i and k to decide which reflector should be open and which streams should be sent to a reflector. The results from rounding stage 1 can be summarized in the following lemma:

LEMMA 5.1. *([2]) The first-stage rounding algorithm incurs a cost at most a factor of* $64 \log |D|$ *higher than the optimum cost, and with high probability violates the weight constraints by at most a factor of* $\frac{1}{4}$ *and the fanout constraints by at most a factor of* 2. *Color constraints are all satisfied.*

By incurring a factor of $\Theta(\log n)$ in the cost, the constant factors loss in the weights and fanouts can be improved as shown in [2].

Second stage rounding: Rounds $x_{i,j,k}$'s using the open reflectors and streams that are sent to different reflectors in the first stage. The results in this stage can be summarized as follows:

LEMMA 5.2. *([2]) The second-stage rounding incurs a cost at most a factor of* 14 *higher than the optimum cost and violates each of fanout, color and weight constraint by at most a factor of* 7.

Our main contribution is an improvement of the second-stage rounding through the use of repeated **RandMove** and by judicious choices of constraints to drop. Let us call the linear program that remains just at the end of first stage **LP-Color2**:

$$\min \sum_{i \in R} \sum_{k \in S} \sum_{j \in D} c_{i,j,k} x_{i,j,k}$$

s.t.

$$\sum_{k \in S} \sum_{j \in D} x_{i,j,k} \leq F_i \ \forall i \in R \text{ (Fanout)}$$

$$\sum_{i \in R} x_{i,j,k} w_{i,j,k} \geq W_{j,k} \ \forall j \in D, \forall k \in S \text{ (Weight)}$$

$$\sum_{i \in R_l} x_{i,j,k} \leq 1 \ \forall j \in D, \ \forall k \in S, \ \forall l \in [m] \text{ (Color)}$$

$$x_{i,j,k} \in \{0,1\} \ \forall i \in R, \forall j \in D, \forall k \in S$$

We show:

LEMMA 5.3. ***LP-Color2*** *can be efficiently rounded such that cost and weight constraints are satisfied exactly, fanout constraints are violated at most by additive* 1 *and color constraints are violated at most by additive* 3.

Proof. Let $x^*_{i,j,k} \in [0,1]$ denote the fraction of stream generated from source $k \in S$ reaching destination $j \in D$ routed through reflector $i \in R$ after the first stage of rounding. Initialize $X = x^*$. The algorithm consists of several iterations. the random value at the end of iteration h is denoted by X^h. Each iteration h conducts a randomized update using **RandMove** on the polytope of a linear system constructed from a subset of constraints of **LP-Color2**. Therefore by induction on h, we will have for all (i,j,h) that $\mathsf{E}\left[X^h_{i,j}\right] = x^*_{i,j}$. Thus the cost constraint is maintained exactly on expectation. The entire procedure can be derandomized giving the required bounds on the cost.

Let R and SD denote the set of reflectors and source, destination pairs respectively. Suppose we are at the beginning of some iteration $(h+1)$ of the overall algorithm and currently looking at the values $X^h_{i,j,k}$. We will maintain two invariants:
(I1") Once a variable $x_{i,j,k}$ gets assigned to 0 or 1, it is never changed;
(I2") Once a constraint is dropped in some iteration, it is never reinstated.
Iteration $(h+1)$ of rounding consists of three main steps:

1. Since we aim to maintain (**I1**), let us remove all $X^h_{i,j,k} \in \{0,1\}$; i.e., we project X^h to those coordinates (i,j,k) for which $X^h_{i,j,k} \in (0,1)$, to obtain the current vector Y of floating (yet to be rounded) variables; let $\mathcal{S} \equiv (A_h Y = u_h)$ denote the current linear system that represents **LP-Color2**. In particular, the fanout constraint for a reflector in $\mathcal{S}$ is its residual fanout F'_i; i.e., F_i minus the number of streams that are routed through it.

2. Let v denote the number of floating variables, i.e., $Y \in (0,1)^v$. We now drop the following constraint:
   (D1") Drop fanout constraint for degree 1 reflector denoted R_1, i.e, reflectors with only one floating variable associated with it. For any degree 2 reflectors denoted R_2, if it has a tight fanout of 1 drop its fanout constraint.
   (D2") Drop color constraint for a group of reflectors R_l, if they have atmost 4 floating variable associated with them.

Let $\mathcal{P}$ denote the polytope defined by this reduced system of constraints. A key claim is that Y is not a vertex of $\mathcal{P}$ and thus we can apply **Rand-Move** and make progress either by rounding a new variable or by dropping a new constraint. We count the number of variables v and the number of tight constraints t separately. We have,

$$t = \sum_{i \in R \setminus R_1} 1 + \sum_{k \in S} \sum_{j \in D} (l_{k,j} + 1),$$

where $l_{j,k}$ is the number of color constraints for the stream generated at source k and to be delivered to the destination j. We have, $v \geq \sum_{i \in R} F_i + 1$. Also the number of variables $v \geq \sum_{k \in S, \mathrm{J} \in D, l_{k,j} > 0} 4 l_{k,j} + \sum_{k \in S, \mathrm{J} \in D, l_{k,j} = 0} 2$. Thus by an averaging argument, the number of variables

$$v \geq \frac{\sum_{i \in R} F_i + 1}{2} + \sum_{k \in S, \mathrm{J} \in D, l_{k,j} > 0} 2 l_{k,j} + \sum_{k \in S, \mathrm{J} \in D, l_{k,j} = 0} 1.$$

A moment's reflection shows that the system can become underdetermined only if there is no color constraint associated with a stream (j,k), each reflector i has two floating variables associated with it with total contribution 1 towards fanout and each stream (j,k) is routed fractionally through two reflectors. But in this situation all the fanout constraints are dropped violating fanout at most by an additive one and making the system underdetermined once again. Coloring constraints are dropped only when there are less than 4 floating variable associated with that group of reflectors. Hence the coloring constraint can be violated at most by an additive factor of 3. The fanout constraint is dropped only for singleton reflectors or degree-2 reflectors with fanout equalling 1. Hence fanout is violated only by an additive 1. Weight constraint is never dropped and maintained exactly. □

6. Future Directions. We discuss two speculative directions related to our rounding approach that appear promising.

Recall the notions of discrepancy and linear discrepancy from the introduction. A well-known result here, due to [12], is that if A is "t-bounded" (every column has at most t nonzeroes), then $\mathrm{lindisc}(A) \leq t$; see [33] for a closely-related result. These results have also helped in the development of improved rounding-based approximation algorithms [9, 49]. A major open question from [12] is whether $\mathrm{lindisc}(A) \leq O(\sqrt{t})$ for any t-bounded matrix A; this, if true, would be best-possible. Ingenious melding of randomized rounding, entropy-based arguments and the pigeonhole principle have helped show that $\mathrm{lindisc}(A) \leq O(\sqrt{t \log n})$ [11, 46, 37], improved further to $O(\sqrt{t \log n})$ in [7]. However, the number of columns n may not be bounded as a function of t, and it would be very interesting to even get some $o(t)$ bound on $\mathrm{lindisc}(A)$, to start with. We have preliminary ideas about using the random-walks approach where the subspace S (that is orthogonal to the set of tight constraints $\mathcal{C}$ in our random-walks approach) has "large" – $\Theta(n)$ – dimension. In a little more detail, whereas the

constraints for rows i of A are dropped in [12] when there are at most t to-be-rounded variables corresponding to the nonzero entries of row i, we propose to do this dropping at some function such as $c_0 t$ to-be-rounded variables, for a large-enough constant c_0 (instead of at t). This approach seems promising as a first step, at least for various models of random t-bounded matrices.

Second, there appears to be a deeper connection between various forms of dependent randomized rounding – such as ours – and iterated rounding [32, 26, 44, 35, 51]. In particular: (i) the result that we improve upon in § 2 is based on iterated rounding [19]; (ii) certain "budgeted" assignment problems that arise in keyword auctions give the same results under iterated rounding [16] and *weighted* dependent rounding [48]; and (iii) our ongoing work suggests that our random-walk approach improves upon the iterated-rounding-based work of [30] on bipartite matchings that are simultaneously "good" w.r.t. multiple linear objectives (this is related to, but not implied by, Theorem 2.1). We believe it would be very fruitful to understand possible deeper links between these two rounding approaches, and to develop common generalizations thereof using such insight.

REFERENCES

[1] A. Ageev and M. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.

[2] K. Andreev, B. Maggs, A. Meyerson, and R. Sitaraman. Designing overlay multicast networks for streaming. In *SPAA*, pages 149–158, 2003.

[3] S. Arora, A. Frieze, and H. Kaplan. A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. *Mathematical Programming*, pages 1–36, 2002.

[4] A. Asadpour, U. Feige, and A. Saberi. Santa Claus meets Hypergraph Matchings. In *Proc. APPROX-RANDOM*, pages 10–20, 2009.

[5] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *STOC '07: Proceedings of the thirty-ninth annual ACM Symposium on Theory of computing*, pages 114–121, 2007.

[6] 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. ACM, 2005.

[7] W. Banaszczyk. Balancing vectors and Gaussian measures of n-dimensional convex bodies. *Random Structures & Algorithms*, 12:351–360, 1998.

[8] N. Bansal and M. Sviridenko. The Santa Claus problem. In *STOC '06: Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 31–40, 2006.

[9] A. Bar-Noy, S. Guha, J. S. Naor, and B. Schieber. Multicasting in heterogeneous networks. In *Proc. ACM Symposium on Theory of Computing*, pages 448–453, 1998.

[10] M. Bateni, M. Charikar, and V. Guruswami. Maxmin allocation via degree lower-bounded arborescences. In *STOC '09: Proceedings of the 41st annual ACM Symposium on Theory of computing*, pages 543–552, 2009.

[11] J. Beck. Roth's estimate on the discrepancy of integer sequences is nearly sharp. *Combinatorica*, 1:319–325, 1981.

[12] J. Beck and T. Fiala. "Integer-making" theorems. *Discrete Applied Mathematics*, 3:1–8, 1981.

[13] J. Beck and V. T. Sós. *Discrepancy theory*, volume II, chapter 26, pages 1405–1446. Elsevier Science B.V. and the MIT Press, 1995.

[14] I. Bezáková and V. D. Allocating indivisible goods. *SIGecom Exch.*, 5(3):11–18, 2005.

[15] D. Chakrabarty, J. Chuzhoy, and S. Khanna. On allocating goods to maximize fairness. In *FOCS '09: 50th Annual IEEE Symposium on Foundations of Computer Science*, 2009.

[16] D. Chakrabarty and G. Goel. On the Approximability of Budgeted Allocations and Improved Lower Bounds for Submodular Welfare Maximization and GAP. In *FOCS*, pages 687–696, 2008.

[17] C. Chekuri and S. Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM J. Comput.*, 35(3):713–728, 2005.

[18] C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding via exchange

properties of combinatorial structures. In *FOCS*, pages 575–584, 2010.

[19] Z. Chi, G. Wang, X. Liu, and J. Liu. Approximating scheduling machines with capacity constraints. In *FAW '09: Proceedings of the Third International Frontiers of Algorithmics Workshop*, pages 283–292, 2009. Corrected version available as arXiv:0906.3056.

[20] J. Chuzhoy and P. Codenotti. Resource minimization job scheduling. In *APPROX*, 2009.

[21] J. Chuzhoy and J. S. Naor. Covering problems with hard constraints. *SIAM Journal on Computing*, 36:498–515, 2006.

[22] T. Ebenlendr, M. Křćal, and J. Sgall. Graph balancing: a special case of scheduling unrelated parallel machines. In *SODA '08: Proceedings of the Nineteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 483–490, 2008.

[23] M. M. Etschmaier and D. F. X. Mathaisel. Airline scheduling: An overview. *Transportation Science*, 19(2):127–138, 1985.

[24] S. Eubank, V. S. A. Kumar, M. V. Marathe, A. Srinivasan, and N. Wang. Structural and algorithmic aspects of massive social networks. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 711–720, 2004.

[25] U. Feige. On allocations that maximize fairness. In *SODA '08: Proceedings of the Nineteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 287–293, 2008.

[26] L. Fleischer, K. Jain, and D. P. Williamson. An iterative rounding 2-approximation algorithm for the element connectivity problem. In *FOCS' 01: Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 339–347, 2001.

[27] R. Gandhi, E. Halperin, S. Khuller, G. Kortsarz, and A. Srinivasan. An improved approximation algorithm for vertex cover with hard capacities. *Journal of Computer and System Sciences*, 72:16–33, 2006.

[28] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding in bipartite graphs. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 323–332, 2002.

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

[30] F. Grandoni, R. Ravi, and M. Singh. Iterative rounding for multi-objective optimization problems. In *ESA '09: Proceedings of the 17th Annual European Symposium on Algorithms*, 2009.

[31] A. Gupta, R. Krishnaswamy, A. Kumar, and D. Segev. Scheduling with outliers. In *Proc. APPROX*, 2009. Full version available as arXiv:0906.2020.

[32] K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21:39–60, 2001.

[33] 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*, 2:113–129, 1987.

[34] V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. A unified approach to scheduling on unrelated parallel machines. *Journal of the ACM*, 56(5), 2009.

[35] L. C. Lau, J. Naor, M. R. Salavatipour, and M. Singh. Survivable network design with degree or order constraints. In *STOC '07: Proceedings of the thirty-ninth annual ACM Symposium on Theory of computing*, pages 651–660, 2007.

[36] J. K. Lenstra, D. B. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

[37] J. Matoušek and J. H. Spencer. Discrepancy in arithmetic progressions. *Journal of the American Mathematical Society*, 9:195–204, 1996.

[38] M. Pál, E. Tardos, and T. Wexler. Facility location with nonuniform hard capacities. In *Proc. Forty-Second Annual Symposium on Foundations of Computer Science*, pages 329–338, 2001.

[39] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM J. Comput.*, 26(2):350–368, 1997.

[40] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 86–92, 2000.

[41] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[42] R. Rushmeier, K. Hoffman, and M. Padberg. Recent advances in exact optimization of airline scheduling problems. Technical report, George Mason University, 1995.

[43] D. B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474, 1993.

[44] M. Singh and L. C. Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *STOC '07: Proceedings of the thirty-ninth annual ACM Symposium on*

Theory of computing, pages 661–670, 2007.

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

[46] J. H. Spencer. Six standard deviations suffice. *Transactions of the American Mathematical Society*, 289:679–706, 1985.

[47] A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *IEEE Symposium on Foundations of Computer Science*, pages 588–597, 2001.

[48] A. Srinivasan. Budgeted allocations in the full-information setting. In *Proc. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 247–253, 2008.

[49] A. Srinivasan and C.-P. Teo. A constant-factor approximation algorithm for packet routing and balancing local vs. global criteria. *SIAM Journal on Computing*, 30:2051–2068, 2001.

[50] L. Tsai. Asymptotic analysis of an algorithm for balanced parallel processor scheduling. *SIAM J. Comput.*, 21(1):59–64, 1992.

[51] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[52] G. Woeginger. A comment on scheduling two parallel machines with capacity constraints. *Discrete Optimization*, 2(3):269–272, 2005.

[53] H. Yang, Y. Ye, and J. Zhang. An approximation algorithm for scheduling two parallel machines with capacity constraints. *Discrete Appl. Math.*, 130(3):449–467, 2003.

[54] J. Zhang and Y. Ye. On the Budgeted MAX-CUT problem and its Application to the Capacitated Two-Parallel Machine Scheduling. Technical report.