SIAM J. COMPUT.
Vol. 28, No. 2, pp. 525–540

PRIMAL-DUAL RNC APPROXIMATION ALGORITHMS FOR SET COVER AND COVERING INTEGER PROGRAMS[*]

SRIDHAR RAJAGOPALAN[†] AND VIJAY V. VAZIRANI[‡]

Abstract. We build on the classical greedy sequential set cover algorithm, in the spirit of the primal-dual schema, to obtain simple parallel approximation algorithms for the set cover problem and its generalizations. Our algorithms use randomization, and our randomized voting lemmas may be of independent interest. Fast parallel approximation algorithms were known before for set cover, though not for the generalizations considered in this paper.

Key words. algorithms, set cover, primal-dual, parallel, approximation, voting lemmas

PII. S0097539793260763

1. Introduction.

Given a universe $\mathcal{U}$, containing n elements, and a collection, $\mathcal{S} \doteq \{S_i : S_i \subseteq \mathcal{U}\}$, of subsets of the universe, the *set cover* problem asks for the smallest subcollection $\mathcal{C} \subseteq \mathcal{S}$ that covers all the n elements in $\mathcal{U}$ (i.e., $\bigcup_{S\in\mathcal{C}} S = \mathcal{U}$). In a more general setting, one can associate a cost, c_S, with each set $S \in \mathcal{S}$ and ask for the minimum cost subcollection which covers all of the elements.[1] We will use m to denote $|\mathcal{S}|$.

Set multicover and *multiset multicover* are successive natural generalizations of the set cover problem. In both problems, each element e has an integer coverage requirement r_e, which specifies how many times e has to be covered. In the case of multiset multicover, element e occurs in a set S with arbitrary multiplicity, denoted $m(S, e)$. Setting $r_e = 1$ and choosing $m(S, e)$ from $\{0, 1\}$ to denote whether S contains e gives back the set cover problem.

The most general problems we address here are *covering integer programs.* These are integer programs that have the following form:

$$\text{MIN } \mathbf{c} \cdot \mathbf{x}, \text{ s.t. } M\mathbf{x} \geq \mathbf{r}, \mathbf{x} \in \overline{\wedge}^{+};$$

the vectors $\mathbf{c}$ and $\mathbf{r}$ and the matrix M are all nonnegative rational numbers.

Because of its generality, wide applicability, and clean combinatorial structure, the set cover problem occupies a central place in the theory of algorithms and approximations. Set cover was one of the problems shown to be NP-complete in Karp's seminal paper [Ka72]. Soon after this, the natural greedy algorithm, which repeatedly adds the set that contains the largest number of uncovered elements to the cover, was shown to be an H_n factor approximation algorithm for this problem ($H_n = 1 + 1/2 + \ldots + 1/n$) by Johnson [Jo74] and Lovasz [Lo75]. This result was extended to the minimum cost case by Chvatal [Ch79]. Lovasz establishes a slightly stronger statement, namely, that the ratio of the greedy solution to the optimum fractional solution is at most

[*] Received by the editors November 26, 1993; accepted for publication (in revised form) October 21, 1996; published electronically July 28, 1998.

http://www.siam.org/journals/sicomp/28-2/26076.html

[†] DIMACS, Princeton University, Princeton, NJ 08544. Current address: IBM Almaden Research Center, San Jose, CA 95120 (sridhar@almaden.ibm.com). This research was done while the author was a graduate student at the University of California, Berkeley, supported by NSF PYI Award CCR 88-96202 and NSF grant IRI 91-20074. Part of this work was done while the author was visiting IIT, Delhi.

[‡] College of Computing, Georgia Institute of Technology, Atlanta, GA (vazirani@cc.gatech.edu). This research was supported by NSF grant CCR-9627308.

[1] Here it is to be understood that the cost of a subcollection $\mathcal{C}$ is $\sum_{S\in\mathcal{C}} c_S$.

H_n. Consequently, the *integrality gap*, the ratio of the optimum integral solution to the optimum fractional one, is at most H_n. An approximation ratio of $O(\log n)$ for this problem has been shown to be essentially tight by Lund and Yannakakis [LY93].[2] More recently, Feige [Fe96] has shown that approximation ratios better than $\ln n$ are unlikely.

The first parallel algorithm for approximating set cover is due to Berger, Rompel, and Shor [BRS89], who found an RNC^5 algorithm with an approximation guarantee of $O(\log n)$. Further, this algorithm can be derandomized to obtain an NC^7 algorithm with the same approximation guarantee. Luby and Nisan [LN93], building on the work of Plotkin, Shmoys, and Tardos [PST91] have obtained a $(1+\epsilon)$ factor (for any constant $\epsilon > 0$), NC^3 approximation algorithm for covering and packing linear programs. Since the integrality gap for set cover is H_n, the Luby–Nisan algorithm approximates the cost of the optimal set cover to within a $(1+\epsilon)H_n$ factor. Furthermore (as noted by Luby and Nisan), in the case of set cover, by using a randomized rounding technique (see [Ra88]), fractional solutions can be rounded to integral solutions at most $O(\log n)$ times their value.

This paper describes a new RNC^3, $O(\log n)$ approximation algorithm for the set cover problem. This algorithm extends naturally to RNC^4 algorithms for the various extensions of set cover each achieving an approximation guarantee of $O(\log n)$. In addition, the approximation guarantee that we obtain in the case of covering integer programs is better than the best known sequential guarantee, due to Dobson [Do82].

2. A closer look at set cover. We begin by taking a closer look at the greedy algorithm for set cover. In the minimum cost case, the greedy algorithm chooses the set that covers new elements at the lowest average cost. More precisely, let $U(S)$ denote the set of yet uncovered elements in S. The greedy algorithm repeatedly adds the set $\mathrm{argmin}_S \left\{ \frac{c_S}{|U(S)|} \right\}$ to the set cover.

Choose $\mathrm{cost}(e)$ to be the cost of covering e by the greedy algorithm. Thus, if S is the first set to cover e and $U(S) = k$ before S was added to the cover, then $\mathrm{cost}(e) \doteq c_S/k$. Let e_i be the ith last element to be covered. Let OPT denote the optimum set cover as well as its cost.

We now observe that $\mathrm{cost}(e_i) \leq \mathrm{OPT}/i$ because there is a collection of sets, namely OPT, which covers all i elements $\{e_1, \ldots, e_i\}$. Thus, there is a set S such that $c_S \leq \mathrm{OPT}/i|U(S)|$. Since e_i is the element of smallest cost among $\{e_1 \cdots e_i\}$, $\mathrm{cost}(e_i) \leq \mathrm{OPT}/i$. Therefore, the cost of the cover obtained by the greedy algorithm is at most $\sum_i \mathrm{cost}(e_i) \leq \mathrm{OPT} \sum_i \frac{1}{i} = \mathrm{OPT} H_n$, which provides the approximation guarantee.

The simple proof given above can be viewed in the more general and powerful framework of linear programming and duality theory. One can state the set cover problem as an integer program:

$$\textbf{IP}: \quad \begin{array}{ll} \text{MIN} & \sum_S c_S x_S \\ \text{s.t.} & \sum_{S \ni e} x_S \geq 1, \\ & x_S \in \{0, 1\}. \end{array}$$

By relaxing the integrality condition on $\mathbf{x}$, we obtain a linear program which has the

[2] More precisely, Lund and Yannakakis establish that there is a constant c such that unless $\tilde{\mathrm{P}} = \tilde{\mathrm{NP}}$, set cover cannot be approximated to a ratio smaller than $c \log n$.

following form and dual:

$$\begin{array}{llcll} \textbf{LP}: & & \qquad & \textbf{DP}: & \\ \text{MIN} & \sum_S c_S x_S & & \text{MAX} & \sum_e y_e \\ \text{s.t.} & \sum_{S \ni e} x_S \geq 1, & & \text{s.t.} & \sum_{e \in S} y_e \leq c_S, \\ & x_S \geq 0; & & & y_e \geq 0. \end{array}$$

The primal linear program is a covering problem and the dual is a packing problem. We will now reanalyze the greedy algorithm in this context. Define value(e) of an uncovered element as

$$\text{value}(e) \doteq \min_{S \ni e} \frac{c_S}{|U(S)|}.$$

The "value" of an element is a nondecreasing function of time in that it gets bigger as more and more elements get covered and therefore each $U(S)$ gets smaller. The greedy algorithm guarantees that cost$(e) =$ value(e) at the moment e is covered.

Now consider any set $S \in \mathcal{S}$. Let $e_i \in S$ be the ith last element of S to be covered by the greedy algorithm. Then, clearly, cost$(e_i) =$ value$(e_i) \leq c_S/i$ at the moment of coverage. Thus, we establish the following inequality for each set S:

$$\sum_{e \in S} \text{cost}(e) \leq \left(1 + \frac{1}{2} + \cdots + \frac{1}{|S|}\right) c_S.$$

Alternately, if k is the size of the largest set $S \in \mathcal{S}$, then the assignment $y_e = \frac{\text{cost}(e)}{H_k}$ is dual feasible. The dual value for this assignment to $\mathbf{y}$ is $\mathbf{DP} = \sum_e y_e = \frac{1}{H_k} \sum_e \text{cost}(e) = \frac{1}{H_k}$greedy cost. Due to the duality theorem of linear programming, **DP** is a lower bound on the value of OPT. Thus, the greedy algorithm approximates the value of set cover to within a factor of H_k.

3. The key ideas behind our parallel algorithm. The greedy algorithm chooses a set to add to the set cover for which $\sum_{e \in S} \text{value}(e) = c_S$. One of the key ideas in this paper is to find a suitable relaxation of this set selection criterion which guarantees that rapid progress is made but does not degrade the approximation guarantee significantly. This is by no means a new notion. Indeed, it has been used in the context of set cover earlier [BRS89]. However, the way in which the relaxation is made and the resulting parallel algorithms are different from the choices made in [BRS89].

In our algorithm we identify cost-effective sets by choosing those that satisfy the inequality

$$\sum_{e \in U(S)} \text{value}(e) \geq \frac{c_S}{2}.$$

This criterion can be distinguished from the greedy criterion in that elements with different values contribute to the desirability of any set. This weighted mixing of diverse elements and the consequent better use of the dual variables, value(e), appears to lend power to our criterion.

Our relaxed criterion guarantees rapid progress. To see this, consider $\alpha \doteq \min_e \text{value}(e)$. Then, any set for which $\frac{c_S}{|U(S)|} \in [\alpha, 2\alpha]$ qualifies to be picked. Thus, after each iteration, the value of α doubles. This and a preprocessing step will enable us to show rapid progress for our algorithm. However, an algorithm based solely on this relaxed criterion will not approximate set cover well, as exhibited by Example 3.0.1.

EXAMPLE 3.0.1. Let $\mathcal{U} \doteq \{1, 2, \ldots, n\}$. Let $\mathcal{S}$ be the $\binom{n}{2}$ sets of size 2 derived from $\mathcal{U}$. The cost of each set is 1. The optimal cover consists of $\frac{n}{2}$ sets. However, the relaxed criterion chooses all the sets. □

We now address the approximation guarantee. We will be able to assign costs to each element, denoted $\text{cost}(e)$, such that the cost of the set cover is at most $\sum_e \text{cost}(e)$. Further, we will establish that at the moment of coverage, $\text{cost}(e) \leq \mu\text{value}(e)$ for each element e and a suitably chosen constant μ. If, for any algorithm, it is possible to choose element costs such that these conditions are satisfied, then we will say that the algorithm has the *parsimonious accounting* property with parameter μ. It is evident from the analysis of the greedy algorithm detailed earlier that the parsimonious accounting property with parameter μ suffices to establish an approximation guarantee of μH_k where k is the cardinality of the largest set in $\mathcal{S}$.

These observations motivate a straightforward method of relaxing the set selection criterion which, however, fails to achieve our stated goal, namely, fast parallel execution. The relaxation and an instance exhibiting its shortcomings are detailed by Example 3.0.2 below.

EXAMPLE 3.0.2. Number the sets arbitrarily. In each iteration, each uncovered element votes for the lowest numbered set that covers it at the least average cost. Any set with more than $\frac{U(S)}{2}$ votes adds itself to the set cover. It is easily verified that this algorithm has the parsimonious accounting property with parameter $\mu = 2$. However, the algorithm can be forced to execute $\Theta(n)$ iterations on the following input: let $\mathcal{U} = \{u_i, v_j : 1 \leq i, j \leq n\}$. Choose $\mathcal{S} = \{U_i, V_j : 1 \leq i, j \leq n-1\}$, where $U_i = \{u_i, u_{i+1}, v_{i+1}\}$ and $V_j = \{v_j, u_{j+1}, v_{j+1}\}$. Finally, choose the set costs to satisfy $c_{U_{i-1}} > c_{U_i} = c_{V_i} > c_{U_{i+1}}$ for ech i. □

The solution we propose finds a compromise between the two strategies detailed above to achieve both objectives, namely, rapid progress as well as good approximation. The critical extra ingredient used in making this possible is the introduction of randomization into the process.

The primal-dual scheme provides a general framework in which we can search for good and fast approximation algorithms for otherwise intractable problems. In the typical case, the hard problem is formulated as an integer program which is then relaxed to obtain a linear program and its dual. In this context, the algorithm starts with a primal, integral infeasible solution and a feasible, suboptimal dual solution. The algorithm proceeds by iteratively improving the feasibility of the primal and the optimality of the dual while maintaining primal integrality until the primal solution becomes feasible. On termination, the obtained primal integral feasible solution is compared directly with the feasible dual solution to give the approximation guarantee. The framework leaves sufficient room to use the combinatorial structure of the problem at hand: in designing the algorithm for the iterative improvement steps and in carrying out the proof of the approximation guarantee.

The greedy algorithm for set cover can be viewed as an instance in the above paradigm. At any intermediate stage of the algorithm let $y_e = \frac{\text{cost}(e)}{H_n}$ if e has been covered and $\frac{\text{value}(e)}{H_n}$ otherwise. Then, by the arguments presented in section 2, $\mathbf{y}$ is feasible for **DP**. The currently picked sets constitute the primal solution.

4. The parallel set cover algorithm.

Our proposed algorithm for set cover is described in Figure 1. The preprocessing step is done once at the inception of

```
PARALLEL SETCOV

Preprocess.

Iteration:
        For each uncovered element $e$, compute value$(e)$.
        For each set $S$: include $S$ in $\mathcal{L}$ if
(•)             $\sum_{e \in U(S)}$ value$(e) \geq \frac{c_s}{2}$.
        Phase:
                (a) Permute $\mathcal{L}$ at random.
                (b) Each uncovered element $e$ votes for first set $S$ (in the random order)
                    such that $e \in S$.
                (c) If $\sum_{e \text{ votes } S}$ value$(e) \geq \frac{c_S}{16}$, $S$ is added to the set cover.
                (d) If any set fails to satisfy (•), it is deleted from $\mathcal{L}$.
        Repeat until $\mathcal{L}$ is empty.
Iterate until all elements are covered.
```

FIG. 1. *The parallel set cover algorithm.*

the computation and is a technical step which we will explicitly establish in the next section. The purpose of this step is to reduce the range of values of c_S to one wherein the largest and smallest values are at most a polynomial factor from each other.

Notice that value(e) is computed only at the beginning of an iteration and is not updated at the inception of each phase.

4.1. Analysis of PARALLEL SETCOV.

We now present an analysis of the proposed algorithm. We will first consider the approximation guarantee and then the running time. The content of the preprocessing step will be defined with the analysis of the running time.

4.1.1. Approximation guarantee.

The algorithm PARALLEL SETCOV satisfies the parsimonious accounting property with $\mu = 16$. If we choose cost(e) = 16value(e) if e votes for S and S is added to the set cover in the same phase, then it is easily verified that $\sum_e$ cost$(e) \geq$ cost of cover.

4.1.2. Running time.

We will now establish that PARALLEL SETCOV is in RNC^3. In order to do this, we will establish the following three assertions.

1. The algorithm executes $O(\log n)$ iterations.
2. With high probability, $(1 - o(1))$, every iteration terminates after $O(\log nm)$ phases.
3. Each of the steps in Figure 1 can be executed in time $\log nmR$, where R is the length of the largest set cost in bits.

These three assertions imply that PARALLEL SETCOV is in RNC^3. Indeed, it follows that PARALLEL SETCOV runs in $(\log n)(\log nm)(\log nmR)$ time on any standard PRAM or circuit model with access to coins.

The third assertion follows from the parallel complexity of integer arithmetic and parallel prefix computations. The details of these operations can be found in a number of standard texts on parallel computation (see [Le92], for instance).

In order to prove the first assertion, we have to detail the preprocessing step. Define $\beta = \max_e \min_{S \ni e} c_S$. Then, $\beta \leq$ cost of optimal cover $\leq n\beta$. The first inequality is because any cover has to pick some set that contains e^*, the maximizing element for β. The second inequality holds because for any arbitrary element e, there is a set of cost at most β containing e. Thus, there is a cover comprising of all these sets of cost at most $n\beta$.

Thus, any set S such that $c_S \geq n\beta$ could not possibly be in the optimum cover. The preprocessing step eliminates all such sets from consideration. Further, if for any element e there is a set S containing it with cost less than $\frac{\beta}{n}$, then we will add S to the set cover immediately. Since there are at most n elements, at most n sets can be added in this manner. These can all be added in parallel and the total cost incurred is at most an additional β. Since β is a lower bound on the cost of the set cover, the additional cost is subsumed in the approximation.

Thus, we can assume that for each set surviving the preprocessing stage, $c_S \in [\frac{\beta}{n}, n\beta]$. This consequence of the preprocessing stage is a key ingredient in establishing the following lemma.

LEMMA 4.1.1. PARALLEL SETCOV *requires at most $O(\log n)$ iterations.*

Proof. Define $\alpha \doteq \min_e \text{value}(e)$. At any point in the current iteration, consider a set S that has not yet been included in the set cover and such that $c_S \leq 2\alpha|U(S)|$. Then, by the definition of α,

$$\sum_{e \in U(S)} \text{value}(e) \geq |U(S)|\alpha \geq \frac{c_S}{2}.$$

Thus S satisfies (•) and must have satisfied it at the inception of the iteration. Thus $S \in \mathcal{L}$. However, at the end of an iteration, $\mathcal{L}$ is empty. Thus for every remaining S, $c_S \geq 2\alpha|U(S)|$, or alternately, $\frac{c_S}{|U(S)|} \geq 2\alpha$. This ensures that in the next iteration, α increases by a factor of 2. Since α is at least $\frac{\beta}{n^2}$ and is at most $n\beta$, there can be no more than $\log n^3 = 3\log n$ iterations. □

We will now show that the number of phases in every iteration is small with high probability. To this end, we will show the following lemma.

LEMMA 4.1.2. *Consider a fixed iteration, i. The probability that the number of phases in iteration i exceeds $O(\log nm)$ is smaller than $\frac{1}{n^2}$.*

Proof. We will focus our attention only on those sets that are in $\mathcal{L}$. Thus, the precondition $\sum_{e \in U(S)} \text{value}(e) \geq \frac{c_S}{2}$ is imposed on all sets that participate in this proof unless otherwise specified. The proof of this lemma is made via a potential function argument. The potential function is $\Phi \doteq \sum_{S \in \mathcal{L}} |U(S)|$. In other words, it is the number of uncovered element-set pairs in $\mathcal{L}$. In what follows, we will show that Φ decreases by a constant fraction (in expectation) after each phase. Since the initial value of Φ is at most nm, the lemma will follow from standard arguments.

Define the *degree* $\deg(e)$ of an element as

$$\deg(e) \doteq |\{S \in \mathcal{L} : S \ni e\}|.$$

Call a set-element pair $(S, e), e \in U(S)$ *good* if $\deg(e) \geq \deg(f)$ for at least three quarters of the elements in $U(S)$.

Let $e, f \in U(S)$ such that $\deg(e) \geq \deg(f)$. Let N_e, N_f, and N_b be the number of sets that contain e but not f, f but not e, and both e and f, respectively. Thus, prob(e votes S) $= \frac{1}{N_e + N_b}$. The probability that both e and f vote for S is exactly the probability that S is the first among $N_e + N_f + N_b$ sets which is exactly $\frac{1}{N_e+N_f+N_b}$. Since $\deg(e) \geq \deg(f)$ implies that $N_e \geq N_f$ we have

$$\text{prob}\,[f \text{ votes } S \mid e \text{ votes } S] = \frac{\text{prob}\,[e \text{ and } f \text{ vote } S]}{\text{prob}\,[e \text{ votes } S]} \geq \frac{N_e + N_b}{N_e + N_b + N_f} > \frac{1}{2}.$$

The statement above provides the heart of the proof, since it implies that if (S, e) is good, then S should get a lot of votes if e votes for S. Thus, under this condition, S should show a tendency to get added to the set cover. We shall now make this formal.

Noting that for any $f \in U(S)$, $\text{value}(f) \leq \frac{c_S}{|U(S)|}$, we see that for any good (S, e), $\sum_{f \in U(S), \deg(f) > \deg(e)} \leq \frac{c_S}{4}$. Since S satisfies ($\bullet$), we obtain

$$\sum_{f \in U(S), \deg(f) \leq \deg(e)} \text{value}(f) \geq \frac{c_S}{2} - \frac{c_S}{4} = \frac{c_S}{4}.$$

The conditional probability statement above allows us to infer that if (S, e) is good and e votes for S, then the expected value of $\sum_{f \in U(S), f \text{ votes } S} \text{value}(f)$ is at least $\frac{c_S}{8}$. An application of Markov's inequality will show that the probability that S is picked is at least $\frac{1}{15}$.

We will ascribe the decrease in Φ when an element e votes for a set S and S is subsequently added to the set cover, to the set-element pair (S, e). The decrease in Φ that will then be ascribed to (S, e) is $\deg(e)$, since Φ decreases by 1 for each set containing e. Since e voted for only one set, any decrease in Φ is ascribed to only one (S, e) pair. Thus, the expected decrease in Φ, denoted $\Delta\Phi$, is at least

$$\begin{aligned} E(\Delta\Phi) &\geq \sum_{(S,e): e \in U(S)} \text{prob}(e \text{ voted for } S,\ S \text{ was picked}) \cdot \deg(e) \\ &\geq \sum_{(S,\, e) \text{ good}} \big(\text{prob}(e \text{ voted for } S) \times \text{prob}(S \text{ was picked} \mid e \text{ voted for } S) \times \deg(e)\big) \\ &\geq \sum_{(S,\, e) \text{ good}} \frac{1}{\deg(e)} \frac{1}{15} \deg(e) \\ &= \frac{1}{15}(\text{number of good } (S, e) \text{ pairs}). \end{aligned}$$

Finally since at least a quarter of all relevant (S, e) pairs are good, we observe that $E(\Delta\Phi) \geq \frac{1}{60}\Phi$. We recall the following fact from probability theory.

Fact 4.1.3. Let $\{X_t\}$ be a sequence of integer-valued and nonnegative random variables such that $E(X_t - X_{t+1} | X_t = x) \geq cx$ for some constant c. Let $Y \doteq \min_k\{X_k = 0\}$. Then, $\text{prob}(Y > O(\log(pX_0))) \leq p$. Here the asymptotic notation hides the dependence on $\frac{1}{c}$ which is linear. □

Notice that we have just established that the evolution of Φ satisfies the preconditions that allow us to apply Fact 4.1.3. Therefore, choosing $p = \frac{1}{n^2}$, we have our lemma. □

THEOREM 4.1. PARALLEL SETCOV *finds a cover which is at most* $16H_n$ *times the optimal set cover. Further, the total running time is* $O(\log n \cdot \log nm \cdot \log nmR)$ *with probability* $1 - \frac{1}{n}$.

Comment. The constant 16 can be improved to $2(1+\epsilon)$ for any $\epsilon > 0$. This is done by changing the number $\frac{c_S}{16}$ in step (c) to $\frac{c_S}{2(1+\epsilon)}$ and the quantity in the definition of ($\bullet$) to $c_S(1-\epsilon^2)$. □

Comment. The conditional statement is a correlation inequality which we feel should be of independent interest. Generalizing this correlation inequality will be a central issue in our analysis of parallel algorithms for the generalizations of set cover. □

5. Set multicover and multiset multicover. The set multicover problem is a natural generalization of the set cover problem. In this generalization, each element e is associated with a coverage requirement, r_e, which indicates the depth to which e must be covered by any feasible cover. Thus, the set multicover problem can be formulated as an integer program as follows: $\text{MIN} \sum_S c_S x_S$ subject to $\sum_{S \ni e} x_S \geq r_e$ and $x_S \in \{0, 1\}$.

Multiset multicover is the generalization where, in addition to the coverage requirement, each element e appears in any set S with a multiplicity $m(S, e)$. Thus, the integer program is $\text{MIN} \sum_S c_S x_S$ subject to $\sum m(S,e) x_S \geq r_e$ and $x_S \in \{0,1\}$.

On relaxing the integrality requirement on x_S we obtain the following linear program and dual in the case of multiset multicover.

$$\textbf{LP}: \quad \begin{array}{ll} \text{MIN} & \sum_S c_S x_S \\ \text{s.t.} & \sum_S m(S,e) x_S \geq r_e, \\ & -x_S \geq -1, \\ & x_S \geq 0; \end{array} \qquad \textbf{DP}: \quad \begin{array}{ll} \text{MAX} & \sum_e r_e y_e - \sum_S z_S \\ \text{s.t.} & \sum_e m(S,e) y_e - z_S \leq c_S, \\ & z_S \geq 0, \\ & y_e \geq 0. \end{array}$$

In the case of set multicover, $m(S, e)$ is simply the indicator function that takes a value of 1 if $e \in S$ and 0 otherwise. The interesting feature here is the need to explicitly limit the value of x_S to at most 1 and the associated appearance of the dual variables z_S. Notice that in the set cover problem the limit of 1 on the value of x_S was implicit.

In the following discussion, we shall largely restrict our attention to the more general case of multiset multicover. In some places, it is possible to obtain slightly stronger results in the case of set multicover. We will indicate these at appropriate points in the text.

5.1. Greedy algorithms. There is a natural greedy sequential algorithm for multiset multicover. Like the set cover algorithm, this algorithm works by repeatedly picking sets until all the coverage requirements are met. At an intermediate stage of this process, let $r(e)$ be the residual requirement of e. Thus, $r(e)$ is initially r_e and is decremented by $m(S, e)$ each time a set S is added to the cover. Define $a(S, e) \doteq \min\{m(S, e), r(e)\}$ and the set of alive elements in S, $A(S)$ to be the multiset containing exactly $a(S, e)$ copies of any element e if S is not already in the set cover, and the empty set if it is. The greedy algorithm repeatedly adds a set minimizing $\frac{c_S}{|A(S)|}$ to the set cover.

We will extend this notation to multisets in general. Thus, we will denote $|A(S)| \doteq \sum_e a(S, e)$. Let $k = \max_S \sum_e m(S, e)$. Let K denote $\sum_e r_e$. Dobson [Do82] shows that this natural extension of the greedy algorithm to set multicover achieves an approximation ratio of H_K. However, he does this by comparing the cost of the multicover obtained directly to the optimum integral solution of the set multicover problem. In what follows, we will establish an approximation ratio of H_k by comparing the greedy solution to the best possible fractional multicover. Since we can always restrict $m(S,e)$ to at most r_e, $k \leq K$. Thus, this represents a slight improvement over Dobson's result.[3]

When a set S is picked, its cost c_S is ascribed equally to each tuple (e,i) where S covers e for the ith time. Here, i ranges from 1 to r_e. We will say S covers (e,i), in short, to describe this case. Obviously, the cost assigned to each tuple is exactly $\mathrm{cost}(e,i) = \frac{c_S}{|A(S)|}$. Now, we choose $y_e = \frac{\max_i\{\mathrm{cost}(e,i)\}}{H_k} = \frac{\mathrm{cost}(e,r_e)}{H_k}$. If a set S is not picked, let $z_S = 0$, and otherwise let

$$z_S = \frac{\sum_{(e,\,i)\text{ covered by }S}(\mathrm{cost}(e,r_e) - \mathrm{cost}(e,i))}{H_k}.$$

The value of this dual assignment is easily verified to be the cost of the greedy multicover divided by H_k.

LEMMA 5.1.1. $\mathbf{y},\mathbf{z}$ *is dual feasible.*

Proof. First, trivially, both $\mathbf{y}$ and $\mathbf{z}$ are nonnegative. Consider for any S,

$$\begin{aligned}&\left(\sum_e m(S,e)y_e\right) - z_S\\&\quad= \frac{1}{H_k}\left(\sum_e m(S,e)\mathrm{cost}(e,r_e) - \sum_{(e,\,i)\text{ covered by }S}(\mathrm{cost}(e,r_e) - \mathrm{cost}(e,i))\right)\\&\quad= \frac{1}{H_k}\left(\sum_{(e,\,i)\text{ covered by }S}\mathrm{cost}(e,i) + \sum_{e\in S,\text{ not covered by }S}\mathrm{cost}(e,r_e)\right).\end{aligned}$$

We want to view a multiset S as a set which contains $m(S,e)$ copies of element e. Notice that there is a term in the right-hand side of the above expression corresponding to each element copy S. Thus, there are $m(S,e)$ terms corresponding to e. Let us arrange the element copies in S in the reverse order in which they were covered, for instance, if $m(S,e) = 10$ and $m(S,f) = 5$, and suppose $r(e)$ fell to 9 before $r(f)$ fell to 4 before $r(e)$ fell to 8. Then, the ninth copy of e precedes the fifth copy of f which precedes the tenth copy of e in this reverse ordering. Notice that the term corresponding to the jth element in this ordering is at most $\frac{c_S}{j}$. Thus, we have

$$\left(\sum_e m(S,e)y_e\right) - z_S \leq \frac{1}{H_k}\left(\sum_{i=1}^{k}\frac{1}{i}\right)c_S.$$

In other words, the dual constraint corresponding to S is satisfied. Thus, we establish the feasibility of $\mathbf{y}$, $\mathbf{z}$ for the dual problem. □

The consequence of the above arguments is the following theorem.

THEOREM 5.1. *The extended greedy algorithm finds a multiset multicover within an H_k factor of* $\mathbf{LP}^*$.

[3] Recall that in the case of set cover, the approximation factor is logarithmic in the size of the largest set and not just in n, the size of the universe. Similarly, in this case, the ratio is logarithmic in k, which is the "local size," as opposed to K, the "global size" of the problem.

```
Parallel MultMultCov
Set $r(e) = r_e$ for each $e$.
Iteration:
        For each element $e$, compute value$(e) = \min_{S \ni e} \frac{c_S}{|A(S)|}$.
        For each set $S$: include $S$ in $\mathcal{L}$ if
(•)           $\sum_e a(S, e)$value$(e) \geq \frac{c_S}{2}$.
        Phase:
            Initialization:
            (a)  Permute $\mathcal{L}$ at random.
            (b)  Each element $e$ votes for first $r(e)$ copies of itself in the random
                 ordering of $\mathcal{L}$.
            (c)  If $\sum_{e \text{ votes } S}$ value$(e) \geq \frac{c_S}{128}$, $S$ is added to the set cover.
            (d)  Decrement $r(e)$ appropriately. Adjust $a(S, e)$ as required.
                 Delete sets that are picked or fail to satisfy (•) from $\mathcal{L}$.
        Repeat until $\mathcal{L}$ is empty.
Iterate until all elements are covered.
```

Fig. 2. *The parallel multiset multicover algorithm.*

5.2. Parsimonious accounting. More pertinently, the proof implies that the parsimonious accounting principle ensures approximation in the case of multiset multicover as well. By this we mean the following. Define the dynamic quantity value$(e) = \min\{\frac{c_S}{|A(S)|}\}$. Then, as long as we can assign costs cost(e, i) where i ranges from 1 to r_e such that

1. cost$(e, i) \leq \mu$value(e) at the moment that the set S covering (e, i) is picked,
2. $\sum_{(e,\, i)}$ cost$(e, i) \geq$ cost of cover,

then the algorithm approximates the value of the multiset multicover to within μH_k. Here k is the largest set size, i.e., $\max_S \sum_e m(S, e)$. We note that in the case of set multicover, k is at most n, and thus we would have an H_n approximation. Moreover, in many instances, k could be substantially smaller than n.

5.3. Parallel algorithms and analysis. We now outline parallel algorithms for multiset multicover. The parallel multiset multicover algorithm is essentially the same as the set cover algorithm except that with each element we associate a dynamic variable, $r(e)$, initially r_e, which tracks the residual requirement of e. After a random permutation of the candidate sets $\mathcal{L}$ is chosen, each element votes for the first $r(e)$ copies of itself in the sequence.[4] The algorithm is detailed in Figure 2. Notice that we can assume without loss of generality that $r(e) \leq \deg(e)$.

5.4. Analysis. It is easy to see that the algorithm satisfies that parsimonious accounting property with $\mu = 128$. This establishes the approximation ratio.

The number of iterations is bounded by $O(\log mnk)$. As earlier, we will denote the cost of the optimum multiset multicover by $\mathbf{IP}^*$. The proof follows exactly along the lines of the proof for the set cover case. The only change required for proving this is in the definition of the crude estimator β: let $S_1, S_2 \cdots S_m$ be the sets arranged in

[4] For example, $r(e)$ is 4 and $m(S_1, e) = 2$, $a(S_2, e) = 3$, and $a(S_3, e) = 2$. Let the permutation be $S_1 < S_2 < S_3$. Then, e votes for S_1 twice and S_2 twice, casting a total of four votes. If the total number of copies of e in the candidate sets is less than $r(e)$, then e votes for them all.

increasing order of cost. Let β_e be the cost of the set containing the r_eth copy of e, and let $\beta = \max_e \beta_e$. Then, $\beta \leq \mathbf{IP}^* \leq mn\beta$. As before, we can restrict attention to the sets such that $c_S \in [\beta/m, mn\beta]$. Again, it can be easily established that $\min_e \text{value}(e)$ increases by a constant factor after each iteration. Since value(e) for any element is at least β/mnk and at most $mn\beta$, there can be only $O(\log mnk)$ iterations.

Notice that for the special case of set multicover, k is at most n. Thus, the bound is $\log mn$ iterations.

5.4.1. Phases in an iteration. The number of phases required in an iteration is at most $O(\log^2 nm)$. This is established by extending the corresponding lemma for set cover. However, the extension is non-trivial and we shall need some machinery to do this.

First we restrict our attention to the sets in $\mathcal{L}$, i.e., sets that satisfy ($\bullet$). In the following discussion, we will denote the copies of element e in the set S by $e^{(i)}$. Here, i ranges from 1 to $a(S,e)$. We say that $e^{(i)}$ votes for S if $e^{(i)}$ is among the first $r(e)$ copies of e in the random ordering of $\mathcal{L}$.

We will now introduce some notation that will simplify our analysis. We denote by $r(e,i) \doteq r(e) - i + 1$. We denote by $\deg(e,i) \doteq \sum_{S\in\mathcal{L}} \min\{a(S,e), r(e,i)\}$. Notice that from the definitions, it follows that $r(e) = r(e,1)$ and $\deg(e) = \deg(e,1)$. The second since $a(S,e)$ is at most $r(e)$. In general, it is tricky to get a handle on the probability that $e^{(i)}$ obtains a vote for S. However, we shall now show the following lemma which says that the quantity $\frac{r(e,i)}{\deg(e,i)}$ approximates this quantity quite nicely.

LEMMA 5.4.1.

$$\frac{1}{2}\frac{r(e,i)}{\deg(e,i)} \leq \text{prob}(e^{(i)} \textit{ votes } S) \leq 4\frac{r(e,i)}{\deg(e,i)}.$$

Proof. The proof has two parts. The first establishes the upper bound, and the second, the lower bound. For the upper bound we notice that with each permutation of the sets such that $e^{(i)}$ votes S, we can associate at least $\frac{\deg(e,i)}{2r(e,i)} - 2$ permutations such that (e,i) does not vote for S. In order to make this association, consider the notion of a "rotation," exhibited by Figure 3. This figure is to be interpreted as follows: the figure shows two permutations of $\mathcal{L}$. These two permutations differ by a "rotation." The shaded rectangles representing each permutation represent the various sets in $\mathcal{L}$. The length of these rectangles correspond to their contributions to $\deg(e,i)$. Thus, the total length of the set of rectangles representing any permutation of $\mathcal{L}$ is exactly $\deg(e,i)$.

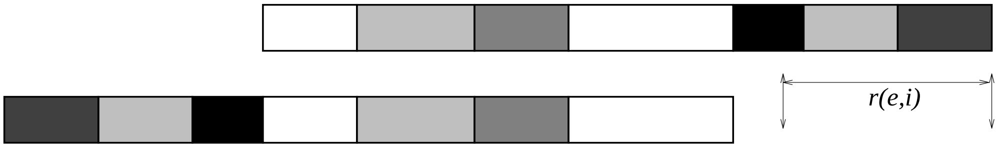

FIG. 3. *The top bar shows a permutation of $\mathcal{L}$, and the bottom, a rotated permutation.*

A rotation is made by extracting from the end of the permutation a minimum number of sets such that they contribute at least $r(e,i)$ towards $\deg(e,i)$ and then placing them in reversed order in the front of the permutation (as shown).

It is easily verified that a rotation is reversible; i.e., it is possible to "unrotate" any rotated permutation to the original permutation. This is most easily seen by turning

Figure 3 upside down. More formally, the unrotation is performed by reversing the sequence, rotating, and then reversing again.

It is also easily verified that for any configuration such that $e^{(i)}$ votes S, it is possible to rotate at least $\frac{\deg(e,i)}{2r(e,i)} - 2$ times such that for each rotated configuration, $e^{(i)}$ does not vote S. This is because the maximum contribution of any set towards $\deg(e,i)$ is $r(e,i)$. Thus, we have associated with each voting permutation at least $\frac{\deg(e,i)}{2r(e,i)} - 2$ nonvoting permutations. Thus, the probability that $e^{(i)}$ votes for S is at most $(\frac{\deg(e,i)}{2r(e,i)} - 1)^{-1} = \frac{2r(e,i)}{\deg(e,i)-2r(e,i)}$. It can be seen via some simple algebraic manipulations that this implies that $\text{prob}(e^{(i)} \text{ votes } S) \leq \frac{4r(e,i)}{\deg(e,i)}$.[5]

For the second part, we need to do some simple analysis. Let us imagine that we permute $\mathcal{L}$ by choosing at random $X_T \in [0,1]$ for each set T and then sorting $\mathcal{L}$ in increasing order of X_T. Notice that we do not need to do this algorithmically; we introduce this just as a means to get a handle on the probability that interests us. Define $Y(x) \doteq \sum_{X_T<x} \min\{a(T,e), r(e,i)\}$. Then, it is easily verified that the events ($e^{(i)}$ votes S) and $(Y(X_S) \leq r(e,i))$ are equivalent. Since for any x, $E(Y(x)) = x \deg(e,i)$, we have by Markov's inequality,

$$\text{prob}(e^{(i)} \text{ votes } S \mid X_S = x) \geq 1 - \text{prob}(Y(x) \geq r(e,i) | X_S = x) \geq 1 - \frac{\deg(e,i)}{r(e,i)} x.$$

Let $\theta \in [0,1]$; then

$$\text{prob}(e^{(i)} \text{ votes } S) \geq \int_0^\theta \text{prob}(e^{(i)} \text{ votes } S \mid X_S = x) dx \geq \theta - \frac{\theta^2}{2} \frac{\deg(e,i)}{r(e,i)}.$$

Choosing $\theta = \frac{r(e,i)}{\deg(e,i)}$ completes the proof, since by our assumption (which, as we pointed out, can be made without loss of generality) that $r(e) \leq \deg(e)$, this fraction is smaller than 1. Otherwise, the lemma is trivial. □

LEMMA 5.4.2. *Let $e^{(i)}, f^{(j)} \in S$. Then*

$$\text{prob}\left(e^{(i)} \textit{ and } f^{(j)} \textit{ vote } S\right) \geq \frac{1}{2} \left(\frac{1}{\frac{\deg(e,i)}{r(e,i)} + \frac{\deg(f,j)}{r(f,j)}} \right).$$

Proof. The proof of this lemma is very similar to the proof of Lemma 5.4.1.

$$\begin{aligned}
&\text{prob}\,(e \text{ and } f \text{ vote } S) \\
&= \int_0^1 \text{prob}\,(e \text{ and } f \text{ vote } S | X_S = x)\, dx \\
&\geq \int_0^\theta \text{prob}\,(e \text{ and } f \text{ vote } S | X_S = x)\, dx, \quad \theta \in [0,1], \\
&\geq \int_0^\theta 1 - \text{prob}\,(e \text{ does not vote } S | X_S = x) - \text{prob}\,(f \text{ does not vote } S | X_S = x)\, dx.
\end{aligned}$$
(1)

[5] To see this, we consider two cases. If $\deg(e,i) \leq 4r(e,i)$, then the conclusion is trivial. Otherwise, it is easy to see that $\frac{2r(e,i)}{\deg(e,i)-2r(e,i)} \leq 4\frac{r(e,i)}{\deg(e,i)}$ by crossmultiplication

Define $Y_e(x)$ and $Y_f(x)$ as in the previous lemma,

$$\text{prob}(e^{(i)} \text{ does not vote for } S \mid X_S = x) \leq \frac{x \cdot \deg(e,i)}{r(e,i)},$$

and do similarly for $f^{(j)}$. Substituting these values back into (1), we obtain

$$\text{prob}(e^{(i)} \text{ and } f^{(j)} \text{ vote for } S) \geq \theta - \left(\frac{\deg(e,i)}{r(e,i)} + \frac{\deg(f,j)}{r(f,j)}\right) \cdot \frac{\theta^2}{2}.$$

Choosing the value of θ to maximize the right-hand side, we get[6]

$$\text{prob}\left(e^{(i)} \text{ and } f^{(j)} \text{ vote } S\right) \geq \frac{1}{2}\left(\frac{1}{\frac{\deg(e,i)}{r(e,i)} + \frac{\deg(f,j)}{r(f,j)}}\right). \qquad \square$$

LEMMA 5.4.3. *Let* $e^{(i)}, f^{(j)} \in S$ *such that* $\frac{r(e,i)}{\deg(e,i)} \leq \frac{r(f,j)}{\deg(f,j)}$.

$$\text{prob}(f^{(j)} \textit{ votes } S \mid e^{(i)} \textit{ votes } S) \geq \frac{1}{8}.$$

Proof. The proof is immediate from the previous two lemmas. □

Remark. In the case of set multicover, the first of the two lemmas implying Lemma 5.4.3 is trivial. Indeed, since $a(S,e)$ is either 0 or 1, we can immediately infer that prob(e votes S) $= \frac{r(e)}{\deg(e)}$. The second lemma (with i and j both set to 1) immediately implies the corresponding version of Lemma 5.4.3 with a bound of $\frac{1}{4}$. □

Say that a set-element pair $(S, e^{(i)})$ is *good* if $\frac{\deg(e,i)}{r(e,i)} \geq \frac{\deg(f,j)}{r(f,j)}$ for at least three quarters of the elements $f^{(j)} \in S$. Then, as in the case of set cover, Lemma 5.4.3 implies that if $(S, e^{(i)})$ is good

$$\text{prob}\left(S \text{ is picked} \mid e^{(i)} \text{ votes for } S\right) \geq p,$$

where $p > 0$. The potential function we use is

$$\Phi \doteq \sum_e \deg(e) H_{r(e)} = \sum_e \deg(e) \sum_{i=1}^{r(e)} \frac{1}{r(e,i)}.$$

Initially, $\Phi \leq mn \log r$, where r is the largest requirement. The expected decrease in Φ ascribable to a good set-element pair $(S, e^{(i)})$, denoted $\Delta\Phi_{(S,e^{(i)})}$, is at least

$$\begin{aligned} E(\Delta\Phi_{(S,e^{(i)})}) &\geq \text{prob}(e^{(i)} \text{ voted for } S) \times \text{prob}(S \text{ was picked} \mid e \text{ voted for } S) \frac{\deg(e)}{r(e,i)} \\ &\geq \frac{r(e,i)}{\deg(e,i)} \frac{p}{2} \frac{\deg(e)}{r(e,i)} \\ &\geq \frac{p}{2}. \end{aligned}$$

[6] Again, the assumption that $r(e) \leq \deg(e)$ implies that our choice of θ is at most 1.

The first inequality is from the definition of Φ, the second is due to Lemmas 5.4.1 and 5.4.3, and the last is due to $\deg(e) \geq \deg(e, i)$. Since a constant fraction of all $(S, e^{(i)})$ are good, $\mathrm{E}\Delta\Phi \geq O(\frac{\Phi}{\log r})$, where r is the largest requirement value. From Fact 4.1.3, we know that the total number of phases is at most $O(\log r(\log mn + \log\log r))$.

THEOREM 5.2. PARALLEL MULTMULTCOV *approximates multiset multicover to within* $128H_n$*, using a linear number of processors and running in time* $O(\log^4 mnr)$.

COROLLARY 5.4.4. *If the number in step* (c) *of Figure* 2 *were changed to* $\frac{c_S}{32}$*, and the algorithm were run on an instance of set multicover, then it would be a* RNC^4 *algorithm approximating set multicover to within* $32H_n$.

Proof. The proof follows from the remark following Lemma 5.4.3. □

6. Covering integer programs.

Covering integer programs are integer programs of the following form:

$$\begin{array}{lll}\textbf{CIP}: & & \\ & \text{MIN} & \sum_S c_S x_S \\ & \text{s.t. } \forall e & \sum_S M(S,e) x_S \geq r_e, \\ & & x_S \in \overline{\wedge}^+.\end{array}$$

Here, $\overline{\wedge}^+$ are the nonnegative integers. The vectors $\mathbf{c}$ and $\mathbf{r}$ and the matrix M are all composed of nonnegative (rational) numbers. At this stage, the notion of a "set" does not have any meaning, however, we continue to use this notation simply to maintain consistency with the previous discussion. For the purpose of the subsequent discussion, the reader should keep in mind that S varies over one set of indices, and e, over the other. Without loss of generality, we may assume that $M(S, e) \leq R_e$.

What we present here is a scaling and rounding trick. The goal is to reduce to an instance of multiset multicover with polynomially bounded (and integral) element multiplicities and requirements. Moreover, in making this reduction, there should be no significant degradation in the approximation factor.

LEMMA 6.0.5. *There is an* NC^1 *reduction from covering integer programs to multiset multicover with element multiplicities and requirements at most* $O(n^2)$ *such that the cost of the fractional optimal (i.e., the cost of the LP relaxation) goes up by at most a constant factor.*

Proof. Essentially, we need to reduce the element requirements to a polynomial range and ensure that the requirements and multiplicities are integral. Then, replicating sets to the extent of the largest element requirement, we would get an instance of multiset multicover. We will achieve this as follows: we will obtain a (crude) approximation to the fractional optimal within a polynomial factor, and then we will ensure that the costs of sets are in a polynomial range around this estimate. Also, we will set all element requirements at a fixed polynomial and set the multiplicities accordingly. At this point, rounding down the multiplicities will change the fractional optimal by only a constant factor.

Let $\beta_e = R_e \cdot \min_S \frac{c_S}{M(S,e)}$ and $\beta = \max_e \beta_e$. Clearly, $\beta \leq \mathbf{CLP}^* \leq n\beta$, where $\mathbf{CLP}^*$ is the cost of the optimal solution to the LP relaxation of $\mathbf{CIP}$. If a set S has large cost, i.e., $c_S > n\beta$, then S will not be used by the fractional optimal, and we will eliminate it from consideration. So, for the remaining sets, $c_S \leq n\beta$. Define $\alpha_S = \lceil \frac{\beta}{nc_S} \rceil$. We will clump together α_S copies of S (i.e., $M(S, e) \leftarrow M(S, e)\alpha_S$, $c_S \leftarrow c_S\alpha_S$). The cost of the set so created is at least $\frac{\beta}{n}$. Additionally, the fractional optimum is not affected by this scaling (though the same cannot be said of the integral optimal). Thus, we can assume that for each S, $c_S \in [\frac{\beta}{n}, n\beta]$.

If any element is covered to its requirement by a set of cost less than $\frac{\beta}{n}$, cover the element using that set and eliminate the element from consideration. The cost incurred in the process is at most β for all elements so covered, and this is subsumed in the constant factor. Notice that as a result of this step, the multiplicity of an element in a set will still be less than its requirements. The reason we require α_S to be integral is that we need to map back solutions from the reduced problem to the original problem. Henceforth, we can assume that the costs satisfy $c_S \in [\frac{\beta}{n}, n\beta]$. Next, we will fix the requirement of each element to be $2n^2$, and we will set the multiplicities appropriately, $m'(S,e) = \frac{M(S,e)}{r_e} \cdot 2n^2$. Since this is just multiplying each inequality by a constant, this will not change the (both fractional and integral) optimal solution or value.

Finally, we will round down the multiplicities, $m(S,e) = \lfloor m'(S,e) \rfloor$. We will show that this will increase the fractional optimal by a factor of at most 4. The same cannot be said for the integral optimum. This is the reason why we needed to compare the solution obtained by our approximation algorithms for multiset multicover to the fractional optimum.

Consider an optimal fractional solution to the problem with multiplicities $m'(S,e)$. Since the cost of this solution is at most $n\beta$, and the c_S is at least $\frac{\beta}{n}$, $\sum_S x_S \le n^2$. Consider an element e, and let $\mathcal{S}$ be the collection of all sets S such that $m'(S,e) < 1$. Then, the total coverage of e due to sets in $\mathcal{S}$ is at most n^2. Therefore, the total coverage of e due to the remaining sets is at least n^2. Since for each of these sets, $m(S,e) \ge \frac{1}{2} m'(S,e)$, if we multiply each x_S by 4, element e will be covered to the extent of at least $2n^2$ in the rounded-down problem. The lemma follows. □

Notice that the reduction in Lemma 6.0.5 is such that a feasible solution to the instance of the multiset multicover problem can be mapped back to a feasible solution of the original problem without increasing the cost. Hence we get the following theorem.

THEOREM 6.1. *There is an $O(\log n)$ factor RNC^4 approximation algorithm for covering integer programs requiring $O(n^2 \#(A))$ processors, where $\#(A)$ is the number of nonzero entries in A.*

Since, in the reduction, all element requirements are set to $O(n^2)$, we obtain the processor bound stated above. Further, since we are comparing the performance of our algorithm with the fractional optimal, it follows that the integrality gap for covering integer programs, in which element multiplicities are bounded by requirements, is bounded by $O(\log n)$. It is easy to see that if multiplicities are not bounded by requirements, the integrality gap can be arbitrarily high.

The previous best (sequential) approximation guarantee for covering integer programs was $H_{nmax(A)}$, where $\max(A)$ is the largest entry in A, assuming that the smallest one is 1 [Do82]. Moreover, in that result, the performance of the algorithm given was compared to the integral optimal.

Notice that the multiset multicover problem with the restriction that each set can be picked at most once is not a covering integer program, since we will have negative coefficients. This raises the question of whether there is a larger class than covering integer programs for which we can achieve (even sequentially) an $O(\log n)$ approximation factor.

Acknowledgments. We thank the referees. Their comments greatly improved the presentation of the paper.

REFERENCES

[BRS89] B. Berger, J. Rompel, and P. Shor, *Efficient NC algorithms for set cover with applications to learning and geometry*, in Proc. 30th IEEE Symposium on the Foundations of Computer Science, 1989, pp. 54–59.

[Ch79] V. Chvatal, *A greedy heuristic for the set covering problem*, Math. Oper. Res., 4 (1979), pp. 233–235.

[Do82] G. Dobson, *Worst-case analysis of greedy heuristics for integer programming with nonnegative data*, Math. Oper. Res., 7 (1982), pp. 515–531.

[Fe96] U. Feige, *A threshold of* $\ln n$ *for approximating set cover*, in Proc. 28th ACM Symposium on the Theory of Computing, 1996, pp. 312–318.

[Jo74] D. S. Johnson, *Approximation algorithms for combinatorial problems*, J. Comput. System Sci., 9 (1974), pp. 256–278.

[Ka72] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972, pp. 85–103.

[LN93] M. Luby and N. Nisan, *A parallel approximation algorithm for positive linear programming*, in Proc. 25th ACM Symposium on Theory of Computing, 1993, pp. 448–457.

[Lo75] L. Lovasz, *On the ratio of optimal integral and fractional covers*, Discrete Math., 13, pp. 383–390.

[Le92] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures*, Morgan Kaufman, San Francisco, 1992.

[LY93] C. Lund and M. Yannakakis, *On the hardness of approximating minimization problems*, in Proc. 25th ACM Symposium on Theory of Computing, 1993, pp. 286–293.

[PST91] S. A. Plotkin, D. B. Shmoys, and E. Tardos, *Fast approximation algorithms for fractional packing and covering problems*, in Proc. 32nd IEEE Symposium on the Foundations of Computer Science, 1991, pp. 495–504.

[Ra88] P. Raghavan, *Probabilistic construction of deterministic algorithms: Approximating packing integer programs*, J. Comput. System Sci., 37 (1988), pp. 130–143.