Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs*

Prabhakar Raghavan

IBM T. J. Watson Research Center, Yorktown Heights, New York 10598

We consider the problem of approximating an integer program by first solving its relaxation linear program and then "rounding" the resulting solution. For several packing problems, we prove probabilistically that there exists an integer solution close to the optimum of the relaxation solution. We then develop a methodology for converting such a probabilistic existence proof to a deterministic approximation algorithm. The algorithm mimics the existence proof in a very strong sense.   © 1988 Academic Press, Inc.

MOTIVATIONS AND MAIN RESULTS

Some of the earliest efforts in integer programming involved solving the underlying relaxation linear program and using the solution to try to find the integer optimum. In general, this does not work well [14]. Recently, Aharoni *et al.* [1] studied the relations between the optimum of an integer program and that of its relaxation, for a class of hypergraph matching and covering problems. We consider several *packing integer programs* arising in combinatorial optimization and the design of integrated circuits. In each case we compute the relaxation optimum and use this information to develop an approximation algorithm for the integer program.

In Section 1 we introduce the *lattice approximation problem*. This problem, first studied by Beck and Fiala [3], can be stated informally as follows. We are given a point p in multidimensional space; we are to find a lattice point q (one whose coordinates are all integers) such that the vector $p - q$ has a "small" inner product with every one of a set S of given vectors. Each vector in S can be thought of as a normal to the hyperplane defining a constraint, or the objective, of a linear program. The point p may be thought of as the solution to the relaxation linear program; we wish to find a feasible lattice point that is "nearby." The requirement

---

130

of small inner product says no constraint is violated by too much; we show that this leads to provably good approximation algorithms.

For each of the problems we consider, we first show the *existence* of a provably good approximate solution using the probabilistic method [5]. In Section 2 we show that the probabilistic existence proof can be converted, in a very precise sense, into a deterministic approximation algorithm. To this end we use an interesting "method of conditional probabilities." In Sections 3, 4, and 5 we apply our methods to integer programs arising in packing, routing, and maximum multicommodity flow. Further applications and directions for work are summarized in Section 6.

Throughout this paper our emphasis will be on the quality of approximation achieved by our algorithms, rather than on their exact running times. From the descriptions of our approximation algorithms, it will be clear that they run in polynomial time. The time take to solve the linear program relaxations of the integer programs dominates the net running time theoretically (and, most likely, in practice as well).

1. THE LATTICE APPROXIMATION PROBLEM

We are given an $n \times r$ matrix C in which $c_{ij} \in [0, 1]$ for all i, j; and an r-vector $p = (p_1, ..., p_r)$, where each p_j is a real number. We are to compute an integer vector (lattice point) $q = (q_1, ..., q_r)$ that approximates p "well" in that every coordinate of $C \cdot (p - q)$ is small in absolute value. We wish to bound the *discrepancies*

$$\Delta_i = \left| \sum_{j=1}^{r} c_{ij}(p_j - q_j) \right| \tag{1.1}$$

in terms of the inner-products

$$s_i = \sum_{j=1}^{r} c_{ij} p_j. \tag{1.2}$$

Without loss of generality, we may consider the reals p_j to be in the interval $[0, 1]$—if not, we subtract their integer parts (floors) and consider the fraction that remains. Throughout this paper, we will consider a restricted class of solutions in which the q_j are "rounded" versions of the p_j, i.e., $q_j \in \{0, 1\}$, for all j. Joel Spencer [17] showed that there always *exists* a lattice point such that $\Delta_i \leqslant 6 \sqrt{n}$, for all i; his proof is unfortunately not constructive.

Suppose we set each q_j to 1 with probability p_j, independently of all the other components of q. Let us call this process *randomized rounding*. Each q_j is thus a Bernoulli trial and $E[q_j] = p_j$. Consider the random variable $\Psi_i = \sum_{j=1}^{r} c_{ij} q_j$:

$$E[\Psi_i] = \sum_{j=1}^{r} c_{ij} E[q_j] = s_i. \tag{1.3}$$

Note that $\Delta_i = |\Psi_i - s_i|$.

1.1. *The Weighted Sum of Bernoulli Trials*

In order to prove existence results using randomized rounding, we require an additional fact from probability theory. We now derive bounds on the tail of the distribution of the weighted sum of Bernoulli trials; these bounds were derived jointly with Joel Spencer. The principles used in their derivation will be useful in the construction of a deterministic algorithm for the lattice approximation problem, in Section 2. Our bounds generalize and improve on bounds on the (unweighted) sum of Bernoulli trials due to Angluin and Valiant [2].

Let a_1, a_2, ..., a_r be reals in $(0, 1]$. Let X_1, X_2, ..., X_r be independent Bernoulli trials with $E[X_j] = p_j$. We wish to study the random variable $\Psi = \sum_{j=1}^{r} a_j X_j$:

$$E[\Psi] = \sum_{j=1}^{r} a_j p_j = m. \tag{1.4}$$

We prove a Chernoff-type bound [4] on the deviations of Ψ above its mean.

THEOREM 1. *Let* $\delta > 0$, *and* $m = E[\Psi] = {} > 0$. *Then*

$$\Pr[\Psi > (1 + \delta) m] < \left[\frac{e^{\delta}}{(1 + \delta)^{(1 + \delta)}} \right]^{m}. \tag{1.5}$$

Proof.

$$\Pr[\Psi > (1 + \delta) m] = \Pr[e^{t\Psi} > e^{t(1 + \delta) m}]$$
$$< e^{-t(1 + \delta) m} E[e^{t\Psi}] \tag{1.6}$$

for any positive real t. The inequality is strict since m and δ both exceed zero. Since the X_j are independent, this can be written as

$$e^{-t(1 + \delta) m} \prod_{j=1}^{r} [p_j e^{t a_j} + 1 - p_j] \leqslant e^{-t(1 + \delta) m} \prod_{j=1}^{r} \exp[p_j(e^{t a_j} - 1)]. \tag{1.7}$$

For $t = \ln(1 + \delta)$, this becomes

$$(1 + \delta)^{-(1 + \delta) m} \exp\left[\sum_{j=1}^{r} p_j \{(1 + \delta)^{a_j} - 1\} \right]$$

$$\leqslant (1 + \delta)^{-(1 + \delta) m} \exp\left[\sum_{j=1}^{r} \delta a_j p_j \right] \tag{1.8}$$

$$= \left[\frac{e^{\delta}}{(1 + \delta)^{(1 + \delta)}} \right]^{m}. \quad \blacksquare \tag{1.9}$$

Similarly, we can prove a theorem regarding deviations of Ψ *below* its mean m.

THEOREM 2. *For* $\gamma \in (0, 1]$,

$$\Pr[\Psi - m < -\gamma m] < \left[\frac{e^{\gamma}}{(1 + \gamma)^{(1 + \gamma)}} \right]^{m}. \tag{1.10}$$

Remark. It is actually possible to give a somewhat tighter bound of $\exp(-\gamma^2 m/2)$ (as did Angluin and Valiant for the case when all $a_j = 1$), but the bound here will be sufficient (and convenient) for our purposes.

By Theorems 1 and 2, we have bounded the tails of the distribution of Ψ by a function that is symmetric in the deviation. This enables us to make the following definitions. We denote by $B(m, \delta)$ our *bound* on the probability that the weighted sum of Bernoulli trials with expectation m exceeds $(1 + \delta) m$, for positive δ:

$$B(m, \delta) = [e^{\delta}/(1 + \delta)^{(1 + \delta)}]^m. \tag{1.11}$$

We denote by $D(m, x)$ the deviation that results in the bound on the tail probability being x:

$$B(m, D(m, x)) = x. \tag{1.12}$$

To give some intuition about the function $D(m, x)$, we consider the following cases.

Case 1. $m > \ln 1/x$. It can be shown that

$$D(m, x) \leqslant (e - 1) \left[\frac{\ln 1/x}{m} \right]^{1/2}. \tag{1.13}$$

Case 2. $m \leqslant \ln 1/x$. A little manipulation yields

$$D(m, x) \leqslant \frac{e \ln 1/x}{m \ln [(e \ln 1/x)/m]}. \tag{1.14}$$

We have thus bounded the deviations above the mean necessary to ensure that the tail probability is bounded by x. By Theorem 2, these bounds also hold for deviations below the mean.

1.2. The Existence Proof

THEOREM 3. *There exists an integer approximation vector q such that*

$$\Delta_i \leqslant s_i D(s_i, 1/2n). \tag{1.15}$$

Proof. We will show that if the integers q_j are selected using randomized rounding, the resulting vector will satisfy (1.15) with non-zero probability. We thus establish the existence of such a q using the *probabilistic method* [5].

Let us say the ith bad event β_i occurs if Δ_i exceeds the bounds of (1.15). Consider

the random variable Ψ_i. By (1.3), its mean is $\sum_{j=1}^{r} c_{ij} p_j = s_i$. By the definitions above,

$$\Pr[\Psi_i > s_i + s_i D(s_i, 1/2n)] < 1/2n$$

$$\Pr[\Psi_i < s_i - s_i D(s_i, 1/2n)] < 1/2n.$$

Thus the probability of bad event β_i is $< 1/n$. Let us say a vector q is "good" if no bad event occurs. Since there are n possible bad events β_i, the probability that the vector produced by randomized rounding is not good is $< n(1/n) = 1$. Thus a randomly chosen vector q is good with non-zero probability, and the theorem follows. ∎

2. The Method of Conditional Probabilities

We now show that the probabilistic existence proof of Theorem 3 can be converted to a deterministic construction of a good vector q. We use an interesting "method of conditional probabilities"; the deterministic algorithm will mimic the probabilistic existence proof in a very strong sense.

It is instructive to model the computation by means of a decision tree. Consider a complete binary tree T of r levels. Level j of T represents the setting of q_j to 0 or 1. For instance, if q_1 is set to 1, we proceed from the root of T to its left son; if q_1 is set to 0, we proceed to the right son. Thus, assigning the variables $q_1, q_2, ...$ in sequence to 0 or 1 amounts to walking down T from the root to a leaf. Each leaf corresponds to one of the 2^r possible vectors q. In terms of the bounds of Theorem 3, we could then speak of "good" leaves and "bad" leaves. Randomized rounding is equivalent to taking the left son at level j with probability p_j, and the right son with probability $1 - p_j$; the choices at the various levels are made independently. Theorem 3 tells us that T always has a good leaf. Our task is to walk down the tree to a good leaf in deterministic polynomial time.

At a typical stage of the computation, we are at some node at level j in the tree, $1 \leqslant j \leqslant r$. We have already walked down the first $j - 1$ levels, assigning $q_1, ..., q_{j-1}$ in the process. We now wish to proceed to one of the two sons of the current node (i.e., assign q_j). Suppose (although this will not be the case) that randomized rounding were executed at levels j through r. Let $P_j(q_1, ..., q_{j-1})$ denote the conditional probability of a bad event occurring given $q_1, ..., q_{j-1}$ and assuming that randomized rounding is used to compute $q_j, ..., q_r$. Then

$$P_j(q_1, ..., q_{j-1}) = p_j P_{j+1}(q_1, ..., q_{j-1}, 1)$$

$$+ (1 - p_j) P_{j+1}(q_1, ..., q_{j-1}, 0) \Rightarrow$$

$$P_j(q_1, ..., q_{j-1}) \geqslant \min\{P_{j+1}(q_1, ..., q_{j-1}, 1), P_{j+1}(q_1, ..., q_{j-1}, 0)\}. \tag{2.1}$$

The following algorithm then suggests itself: for $j = 1$ to r, at level j we set q_j to 0

or 1 so as to minimize $P_{j+1}(q_1, ..., q_{j-1}, q_j)$. The existence of at least one good leaf (Theorem 3) implies that $P_1 < 1$; combining this inductively with Eq. (2.1), we conclude that

$$1 > P_1 > P_2(q_1) > P_3(q_1, q_2) > \cdots > P_r(q_1, ..., q_{r-1}) > P(\text{leaf}), \qquad (2.2)$$

where $P(\text{leaf})$ is the probability that we have reached a bad leaf. Every leaf is either bad or good; accordingly, $P(\text{leaf})$ is either 0 or 1. But our procedure takes us to a leaf for which $P(\text{leaf}) < 1$, so $P(\text{leaf})$ must be 0 and the leaf we have reached must be good.

From an algorithmic standpoint, the difficulty lies in computing these conditional probabilities efficiently. Let $U_j(q_1, ..., q_{j-1})$ be an upper bound on $P_j(q_1, ..., q_{j-1})$ for all j, that can be *efficiently computed*. Further, let $U_j(q_1, ..., q_{j-1})$ have the property that

$$U_j(q_1, ..., q_{j-1}) \geqslant \min\{U_{j+1}(q_1, ..., q_{j-1}, 1), U_{j+1}(q_1, ..., q_{j-1}, 0)\}. \qquad (2.3)$$

Our algorithm would then be: for $j = 1$ to r, assign to q_j that value which minimizes $U_{j+1}(q_1, ..., q_{j-1}, q_j)$. At each stage:

(a)   the function U is an upper bound on the function P (temporarily omitting subscripts, etc. for brevity);

(b)   By (2.3), U never rises in the course of the computation;

(c)   the algorithm can be run efficiently since U can be computed efficiently. We call this the *method of pessimistic estimators*, since at each stage we bound the probability of failure from above. If we could find a pessimistic estimator such that $U(\text{root}) < 1$, we are guaranteed to succeed.

2.1. Moment-Generating Functions and the Function U

We now derive a suitable function U; the manner in which we do so parallels the proofs of the bounds in Theorems 1 and 2, and the existence proof of Theorem 3. Recall that we said that the ith bad event β_i is said to occur if, for the vector q that we compute, the ith discrepancy Δ_i exceeds the limits prescribed by Theorem 3. Let

$$L_{i+} = s_i[1 + D(s_i, 1/2n)]$$
$$L_{i-} = s_i[1 - D(s_i, 1/2n)]. \qquad (2.4)$$

Thus, bad even β_i occurs when $\Psi_i > L_{i+}$ or $\Psi_i < L_{i-}$.

2.1.1. Bounding the Probability of β_i at the Beginning

Consider the probability of bad event β_i resulting from Ψ_i exceeding L_{i+}, at the beginning of the computation (at the root of T). Following (1.7), for any real $t_i \geqslant 0$

$$\Pr[\Psi_i > L_{i+}] < e^{-t_i L_{i+}} \prod_{j=1}^{r} [p_j e^{c_{ij} t_i} + 1 - p_j]. \qquad (2.5)$$

2.1.2. *Updating the Bound: The Effect of Setting q_k to 0 or 1*

Suppose some q_k were assigned the value 1. Given this information, the conditional probability that Ψ_i exceeds L_{i+} is the probability that the sum of the remaining random variables exceeds $L_{i+} - c_{ik}$. This is bounded above by

$$e^{-t_i(L_{i+} - c_{ik})} \prod_{j \neq k} E[e^{t_i c_{ij} q_j}] = e^{-t_i L_{i+} + c_{ik} t_i} \prod_{j! = k} [p_j e^{c_{ij} t_i} + 1 - p_j].$$

Thus the conditional probability of Ψ_i exceeding L_{i+} given $q_k = 1$ is just bounded by replacing the term

$$p_k e^{c_{ik} t_i} + 1 - p_k$$

by $e^{c_{ik} t_i}$ in the bound function—an intuitively correct idea. Likewise, it can be verified that setting $q_k = 0$ has the effect that the term

$$p_k e^{c_{ik} t_i} + 1 - p_k$$

is replaced by 1.

2.1.3. *The function U*

The probability that *any one* of the random variables Ψ_i exceeds its upper limit is bounded above by the sum of the individual probabilities in (2.5):

$$\sum_{i=1}^{n} e^{-t_i L_{i+}} \prod_{j=1}^{r} [p_j e^{c_{ij} t_i} + 1 - p_j]. \tag{2.7}$$

So far, we have discussed deviations of the random variables Ψ_i *above* their means; a similar analysis gives a bound on the probability that for some i, Ψ_i falls *below* its lower limit L_{i-}. Adding this bound to (2.7), we obtain an upper bound on the probability that *any* bad event β_i occurs:

$$U(\text{root}) = \sum_{i=1}^{n} \left\{ e^{-t_i L_{i+}} \prod_{j=1}^{r} [p_j e^{c_{ij} t_i} + 1 - p_j] + e^{+t_i L_{i-}} \prod_{j=1}^{r} [p_j e^{-c_{ij} t_i} + 1 - p_j] \right\}$$
$$< 1. \tag{2.8}$$

The last inequality holds for $t_i = \ln[1 + D(s_i, 1/2n)]$. Indeed, we used the above bound (through Theorems 1 and 2) in the proof of Theorem 3, with these values of t_i. Equation (2.8) gives us the value of U at the root of T. We saw (Section 2.1.2) the effect of assigning some q_k to 0 or 1; the updated value of U is always an upper bound on the probability of a bad event, conditioned by the assignment of q_k.

It remains to show that for any k, one of the two possible assignments of q_k reduces the value of U. We will show that this property is satisfied by $U(\text{root})$; a

similar argument applies to subsequent stages. We thus examine the effect of setting q_1. Equation (2.8) for U can be written as

$$\sum_{i=1}^{n} B_i(p_1 e^{c_{i1} t_i} + 1 - p_1) + \sum_{i=1}^{n} C_i(p_1 e^{-c_{i1} t_i} + 1 - p_1)$$

$$= p_1 \sum_{i=1}^{n} (B_i e^{c_{i1} t_i} + C_i e^{-c_{i1} t_i}) + (1 - p_1) \sum_{i=1}^{n} (B_i + C_i), \qquad (2.9)$$

where B_i and C_i are fixed numbers. If q_1 is set to 1, the new value of U is

$$\sum_{i=1}^{n} (B_i e^{c_{i1} t_i} + C_i e^{-c_{i1} t_i}) \qquad (2.10)$$

while if q_1 is set to 0 the new value of U is

$$\sum_{i=1}^{n} (B_i + C_i). \qquad (2.11)$$

Since (2.9) is a convex combination of (2.10) and (2.11), it is no less than the smaller of (2.10) and (2.11). Thus we can proceed from the root of T to one of its sons in such a manner that U does not rise. A similar argument for the general step (updating U as we proceed) shows that the value of U does not rise in the course of the computation. Thus $1 > U(\text{leaf}) > P(\text{leaf})$.

THEOREM 4. *The method of pessimistic estimators yields in deterministic polynomial time an integer vector q such that*

$$\Delta_i \leqslant s_i D(s_i, 1/2n), \qquad 1 \leqslant i \leqslant n. \qquad (2.12)$$

This improves on a result of Beck and Fiala [3] who studied the case $c_{ij} = 0$ or 1; for this case they showed an algorithm for constructing an integer approximation q such that $\Delta_i \leqslant (8n \ln 2n)^{1/2}$, for all i. Using (1.13) and (1.14), we find that the discrepancies guaranteed by our algorithm are asymptotically smaller than those of the Beck–Fiala algorithm when s_i is $o(n)$. When s_i grows as n, our constant factors are better. In Sections 3–5 we consider applications of the theory developed above to approximately solving certain integer programs.

2.2 The Effect of the Model of Computation

The function U in (2.8) requires the computation of exponential functions. In practice this could be done by means of suitable approximations, perhaps yielding a vector q with a guarantee close to that of Theorem 4. However, if we allow ourselves only elementary arithmetic operations (as in the RAM model of computation), the function U of (2.8) is not efficiently computable. We now show that our results hold with slight modifications in the RAM model of computation.

We first observe that for our choice $t_i = \ln[1 + D(s_i, 1/2n)]$,

$$e^{t_i} = 1 + D(s_i, 1/2n). \tag{2.13}$$

Thus we do not have to compute logarithms. Examination of (2.8) shows that e^{t_i} (or e^{-t_i}) is raised to the power L_{i-} (or L_{i+}). We ensure that these are integral powers by replacing L_{i-} by $L'_{i-} = \lfloor L_{i-} \rfloor$ (and L_{i+} by $L'_{i+} = \lceil L_{i+} \rceil$). These integers are "small"—they do not exceed r in magnitude. Next, we observe that Theorem 3 holds *a fortiori* if we allow deviations upto $L'_{i+} - s_i$ on the positive side and down to $L'_{i-} - s_i$ on the negative side. Consequently, $U(\text{root})$ remains < 1 when the new values L'_{i+} and L'_{i-} are used in (2.8).

The main obstacle remaining in our computation of U is the exponentiation of e^{t_i} to the power c_{ij}, which we have so far permitted to be an arbitrary real. There does not seem to be an obvious way around this; we thus have to restrict our instances to those for which $c_{ij} \in \{0, 1\}$. Certainly this restriction does not affect our solutions to the packing integer programs in the following sections. Whether we can allow the c_{ij} to assume real values and still achieve similar lattice approximations on the RAM model of computation remains an interesting open problem.

3. VECTOR SELECTION AND ROUTING PROBLEMS

In the *vector selection* problem we are given Λ, a collection of sets of vectors. Let $\Lambda = \{\lambda_1, ..., \lambda_r\}$. Each set λ_j consists of n-vectors $\{V_1^j, ..., V_{k_j}^j\}$, where $k_j = |\lambda_j|$. For $1 \leq i \leq n$, the ith component $v_k^j(i)$ of vector V_k^j is either 0 or 1. We are to choose exactly one vector from each set λ_j; we denote by V^j the vector chosen from λ_j. We wish to minimize $\|\sum_{j=1}^r V^j\|_\infty$. This problem has important applications to global routing in gate-arrays [7, 10, 15]; λ_j represents the possible routes for a set of gates to be connected in an integrated circuit. We wish to choose a route connecting each set of gates so as to minimize the space requirements of the routing; details may be found in [15].

This can be formulated as an integer program as follows. We use an indicator $(0-1)$ variable x_k^j to indicate whether or not the vector V_k^j is selected to represent λ_j; here $1 \leq j \leq r$ and $1 \leq k \leq k_j$. The integer program is then: Minimize W subject to $x_k^j \in 0, 1$ and

$$\sum_{k=1}^{k_j} x_k^j = 1, \qquad\qquad 1 \leq j \leq r$$
$$\tag{3.1}$$
$$\sum_{j=1}^r \sum_{k=1}^{k_j} x_k^j \cdot v_k^j(i) \leq W, \qquad 1 \leq i \leq n.$$

This integer program can be shown to be NP-hard by a reduction very similar to that of Kramer and van Leeuwen [11]. We solve the relaxation linear program with $x_k^j \in [0.1]$; this yields fractional s lutions for the variables in the linear

program which we denote by $\hat{x}_k^j$. Let W' be the value of the objective function. The application of randomized rounding consists of choosing V_k^j to represent λ_j with probability $\hat{x}_k^j$. The choice is made independently for the different j and mutually exclusively among the vectors in λ_j (this may be thought of as casting a λ_j-faced die whose face probabilities are $\hat{x}_k^j$). Using Theorem 1, we cannow prove

THEOREM 5. *There exists a solution to the integer program* (3.1) *with* $W \leqslant W^E$, *where*

$$W^E = W' \cdot [1 + D(W', 1/n)].   \qquad (3.2)$$

Proof. The linear program solutions satisfy

$$\sum_{j=1}^{r} \sum_{k=1}^{k_j} \hat{x}_k^j \cdot v_k^j(i) \leqslant W', \qquad 1 \leqslant i \leqslant n.   \qquad (3.3)$$

For each i, randomized rounding makes (3.3) the sum of independent Bernoulli trials of mean $\leqslant W'$. The use of (1.12) yields the theorem (further details may be found in [15]). ∎

Using the method of pessimistic estimators, we have

THEOREM 6. *We can approximate* (3.1) *in deterministic polynomial time to obtain an integer solution with objective* W^I *such that*

$$W^I \leqslant \lceil W^E \rceil.   \qquad (3.4)$$

Proof. We use the method of conditional probabilities; the decision tree is no longer binary, but rather has k_j branches at level j. We construct a function U; the upper bound from the moment generating function of Ψ_i now has terms of the form

$$e^{-t_i \lceil W^E \rceil} \prod_{j=1}^{r} \left[\sum_{k=1}^{k_j} \hat{x}_k^j \exp[v_k^j(i) t_i] \right]   \qquad (3.5)$$

corresponding to (2.5). The analysis is similar to that leading to Theorem 4. ∎

The value W' is a lower bound on the integer optimum. Using (1.13) and (1.14) with $x = 1/n$, we are guaranteed of finding an integer solution within a multiplicative factor of the optimum; we thus have a *fully polynomial-time approximation scheme* (FPTAS) [14].

We pause to give the reader a concrete example of the kind of performance bound our algorithm delivers. When $W' > \ln n$, we use (1.13) to show that our approximation finds an integer solution with

$$W^I \leqslant \lceil W' + (e-1)[W' \ln n]^{1/2} \rceil,$$

where W' is a lower bound on the best possible solution.

4. PACKING PROBLEMS

Let A be an $n \times r$ matrix in which each entry $a_{ij} \in \{0, 1\}$. Consider the following integer linear program, with $x_j \in \{0, 1\}$:

$$\text{Max} \sum_{j=1}^{r} a_{1j} x_j \quad \text{s.t.} \sum_{j=1}^{r} a_{ij} x_j \leqslant k, \quad 2 \leqslant i \leqslant n. \tag{4.1}$$

This is a packing integer program in the following sense: we are trying to pack as many of the column-vectors of A as possible into an n-dimensional cube of side k. The vector sum of the chosen vectors should fit in the cube. There is a scheduling interpretation to the integer program (4.1). Each of the variables x_j may be associated with a task. Rows 2 through n of the matrix each represent a machine. The entry a_{ij} indicates whether a unit time of machine i is required for the execution of task j. We wish to maximize the number of tasks that can be scheduled for execution within a finishing time k; here $a_{1j} = 1$, $1 \leqslant j \leqslant r$.

For $a_{ij} \in \{0, 1\}$, Lovász [12] calls this problem *simple k-matching* in an $(n-1)$-node hypergraph. Rows 2 through n of the matrix A can be thought of as the incidence matrix of a hypergraph H, with the rows representing the vertices and the columns the edges. The element a_{ij} is a 1 if edge j is incident on vertex i. Again, $a_{1j} = 1$, $1 \leqslant j \leqslant r$. The integer program (4.1) seeks the largest set of edges no more than k of which are incident on any vertex.

In the following description, we use the terminology of k-matching. We solve the relaxation linear program with $x_j \in [0, 1]$. Let the linear program yield a value x_j^* for the variable x_j. Let the value of the objective function be M^*; Lovász [12] calls M^* the *fractional k-matching number* of H. For all j, independently set x_j to 1 with probability x_j^*. Let the resultant rounded value of variable x_j be $x_j^{(r)}$. The difficulty now is that after rounding, $\sum_{j=1}^{r} a_{ij} x_j^{(r)}$ may exceed k for some i, thus violating a constraint. Let $v \in (0, 1)$ be a number such that

$$B\left(vk, \frac{1-v}{v}\right) < \frac{1}{n}. \tag{4.2}$$

We have not as yet established under what conditions such a v must exist; let us for the moment continue under the assumption that we do have such a value of v. The idea is to multiply each x_j^* by v before rounding. Let $x_j^S = v x_j^*$. The superscript S indicates a fractional value that has been scaled. As a result, the fractional value of the objective function is also scaled down by the factor v; we let M^S denote vM^*. Randomized rounding now consists of rounding variable x_j to 1 with probability x_j^S. We now show that there exists a solution such that no constraint is violated *and* the rounded value of the objective function does not fall "too far" below M^S.

THEOREM 7. *There exists a k-matching of cardinality*

$$\geqslant M^S \cdot [1 - D(M^S, 1/n)]. \tag{4.3}$$

Proof. After rounding, the expected value of each constraint is no more than vk. By our choice of v (4.2), the probability that a constraint is violated (i.e., its value exceeds k) is thus less than $1/n$. Thus the proability that *any* constraint is violated is less than $(n-1)/n$. The expected value of the objective function is M^S. The probability that if falls below (4.3) is less than $1/n$. Thus, there is a non-zero probability of a matching of the size given by (4.3). $\blacksquare$

Using the method of pessimistic estimators and the scaled variables x_j^S, we have:

THEOREM 8. *We can compute in deterministic polynomial time a k-matching of cardinality at least*

$$\lfloor M^S \cdot [1 - D(M^S, 1/n]\,\rfloor. \tag{4.4}$$

The relaxation linear program optimum M^* is an upper bound on the integer optimum. The value M^S in theorem 7 is smaller than M^* by the multiplicative factor v. Theorem 7 assures us of finding a k-matching that is smaller than M^S by a subtractive factor.

For what values of k does there exist a positive value of v satisfying (4.2)? Examination of (1.13) reveals that if $k > \ln n$, v is a positive constant. In this case we have a FPTAS which approximates the k-matching to within a contant factor. If $k \leqslant \ln n$, we still obtain a FPTAS by (1.14), though the approximation is not to within a multiplicative constant. Further details of these approximations may be found in [16].

5. MAXIMUM MULTICOMMODITY FLOW

Maximum 0–1 multicommodity flow is an important problem in operations research [14]. We are given a directed graph $G(V, E)$, and k source–sink pairs. Each edge *member* E has a positive capacity $c(e)$. For $1 \leqslant j \leqslant k$, the flow of commodity j is said to be realized if we convey one unit of flow from source s_j to the corresponding sink t_j. The flow must be *integral*; i.e., we must specify a path in G from s_j to t_j. We wish to maximize the number of commodities whose flow is realized (i.e., the total flow), with the constraint that the total flow in any edge e does not exceed $c(e)$.

This problem can be formulated as a 0–1 integer linear program. We know that optimizing this integer linear program is NP-hard [6, 9]; but the relaxation linear program can be solved efficiently. Let $F^{(I)}$ be the optimum integer flow; let $N = |E|$ be the number of edges in the network; and let c be the smallest edge capacity. Let $F^* \geqslant F^{(I)}$ be the fractional maximum flow. Define v as in (4.2):

$$B\left(vc, \frac{1-v}{v}\right) < \frac{1}{N+1}. \tag{5.1}$$

Let F^S be the scaled total flow. Using methods similar to those in Sections 3 and 4, we can show

THEOREM 9.   *Pessimistic estimators will find a total multicommodity flow*

$$\geq \lfloor F^S \cdot [1 - D(F^S, 1/N + 1)] \rfloor. \tag{5.2}$$

6. CONCLUDING REMARKS

We have considered the problem of approximating an integer program by first solving its linear program relaxation and rounding the resulting solution. For each of the problems considered, we first presented a probabilistic proof of the existence of an integer solution close to the linear program optimum. In Section 2 we presented a methodology—the method of pessimistic estimators of conditional probability—for converting such an existence proof into a deterministic approximation algorithm. The vehicle used for developing this methodology was the lattice approximation problem. The lattice approximation problem appears intrinsic to the conversion to the conversion of linear program solutions to approximate integer program solutions. Sections 3–5 outlined applications of our technique to problems of practical interest. The following issues are noteworthy:

(1)   Our algorithms use linear programming as a preliminary phase, before rounding. The entire process is a polynomial-time computation due to the efficient algorithms of Karmarkar [8] and others.

(2)   Our methods improve on algorithms for some combinatorial problems studied by Olsen and Spencer [13]. These involve 2-coloring the vertices of a hypergraph and set-balancing.

(3)   What other randomized procedures can be made deterministic by our methods?

(4)   Our deterministic algorithm is highly sequential, in that we round one variable at a time; is there an efficient way of determinstically rounding in parallel?

(5)   Throughout, we naively (?) sum the probabilities of all bad events—although these bad events are surely correlated. Can we prove a stronger result using algebraic properties (e.g., the rank) of the coefficient matrix? A tighter bound for the probabilistic existence proofs should lead to tighter approximation algorithms. When the sum of the entries in every column of the coefficient matrix is bounded above by some number g, Karp *et al.* [10] give a technique for rounding such that all discrepancies are bounded by g.

REFERENCES

1. R. AHARONI, P. ERDÖS, AND N. LINIAL, Dual integer linear programs and the relationship between their optima, in "Proceedings, 17th ACM Symposium on Theory of Computing, Providence, RI, May 1985, pp. 476–482.
2. D. ANGLUIN AND L. G. VALIANT, Fast probabilistic algorithms for Hamiltonian circuits and matchings, J. Comput. System Sci. 19 (1979), 155–193.
3. J. BECK AND T. FIALA, "Integer-making" theorems, Discrete Appl. Math. 3 (1981), 1–8.
4. H. CHERNOFF, A measure of asymptotic efficiency for tests based on the sum of observations, Ann. Math. Statist. 23 (1952), 493–509.
5. P. ERDÖS AND J. SPENCER, "The Probabilistic Method in Combinatorics," Academic Press, New York/London, 1974.
6. S. EVEN, A. ITAI, AND A. SHAMIR, On the complexity of timetable and multicommodity flow problems, SIAM J. Comput. 5 (1976), 691–703.
7. T. C. HU AND M. T. SHING, A decomposition algorithm for circuit routing, Math. Programming Stud. 24 (1985), 87–103.
8. N. KARMARKAR, A new polynomial-time algorithm for linear programming Combinatorica 4 (1984), 373–396.
9. R. M. KARP, Reducibility among combinatorial problems, in "Complexity of Computer Computations" (R. N. Miller and J. W. Thatcher, Eds.), pp. 85–104, Plenum, New York, 1972.
10. 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, in "Proceedings, 24th Annual Symposium on Foundations of Computer Science, 1983," pp. 453–459.
11. M. R. KRAMER AND J. VAN LEEUWEN, "Wire-Routing is NP-Complete," Technical Report RUU-CS-82-4, Department of Computer Science, Rijksuniversiteit Utrecht, 1982.
12. L. LOVÁSZ, On the ratio of optimal and fractional covers, Discrete Math. 13 (1975), 383–390.
13. J. E. OLSON AND J. SPENCER, Balancing families of sets, J. Combin. Theory Ser. A 25 (1978), 29–37.
14. C. H. PAPADIMITRIOU AND K. STEIGLITZ, "Combinatorial Optimization: Algorithms and Complexity," Prentice-Hall, Englewood Cliffs, NJ, 1982.
15. P. RAGHAVAN AND C. D. THOMPSON, Provably good routing in graphs: Regular arrays, in "Proceedings, 17th ACM Symposium on Theory of Computing, 1985," pp. 79–87.
16. P. RAGHAVAN AND C. D. THOMPSON, Randomized rounding: Provably good algorithms and algorithmic proofs, Combinatorica 7 (1987), 365–374.
17. J. SPENCER, Six standard deviations, Trans. Amer. Math. Soc. 289 (1985), 679–706.