# NEW ALGORITHMIC ASPECTS OF THE LOCAL LEMMA WITH APPLICATIONS TO ROUTING AND PARTITIONING*

TOM LEIGHTON†, CHI-JEN LU‡, SATISH RAO§, AND ARAVIND SRINIVASAN¶

**Abstract.** The Lovász Local Lemma (LLL) is a powerful tool that is increasingly playing a valuable role in computer science. The original lemma was nonconstructive; a breakthrough of Beck and its generalizations (due to Alon and Molloy & Reed) have led to constructive versions. However, these methods do not capture some classes of applications of the LLL. We make progress on this, by providing algorithmic approaches to two families of applications of the LLL. The first provides constructive versions of certain applications of an extension of the LLL (modeling, e.g., hypergraph-partitioning and low-congestion routing problems); the second provides new algorithmic results on constructing disjoint paths in graphs. Our results can also be seen as constructive upper bounds on the integrality gap of certain packing problems. One common theme of our work is a "gradual rounding" approach.

**1. Introduction.** The Lovász Local Lemma (LLL) is a powerful tool that can be used to show the existence of various types of (discrete) structures [7]. Let $e$ denote the base of natural logarithms, and $E_1, E_2, \ldots, E_m$ be events with $\max_i \Pr[E_i] \leq p$. If each $E_i$ is mutually independent of all but at most $D$ of the other events $E_j$ and if $ep(D+1) \leq 1$, then the key claim of the LLL (symmetric case) is that $\Pr[\bigwedge_{i=1}^{m} \overline{E_i}] \geq (1-ep)^m > 0$. See, e.g., [3, 19] for many applications of this. The term $(1-ep)^m$, though positive, is often "tiny"; so the result does not directly imply an efficient algorithm to produce a structure that avoids all the $E_i$. Breakthroughs in this direction have been made by Beck [4], and generalized by Alon [1] and Molloy & Reed [18]. However, as mentioned in [4, 1, 18], certain classes of applications of the LLL are not covered by these results. One such class arises because the approaches of [4, 1, 18] require $pD^3 = \mathrm{O}(1)$; they require something close to "$pD^9 = \mathrm{O}(1)$" for some situations. More importantly, as sketched in Section 4.1, they do not guarantee polynomial-time algorithms in some settings where the underlying random variables have "large" ranges, even if $pD^c = \mathrm{O}(1)$ for an arbitrarily large constant $c$. Another such class involves some extensions of the LLL where $D \gg p^{-1}$. Here, we provide constructive approaches for certain instances of both of these classes; we now start with a framework that motivates many of the problems considered.

Let $[t] \doteq \{1, 2, \ldots, t\}$. The NP-hard *low-congestion routing* problem has attracted much attention, e.g., from the VLSI layout and network routing viewpoints [21, 11]. Given a graph $G = (V, E)$ and a collection $\mathcal{T} = \{(s_i, t_i) : i \in [k]\}$ of pairs of vertices in

*Conference versions of this work appeared in the following two papers, both in the *ACM-SIAM Symposium on Discrete Algorithms*, 1999: (i) "New algorithmic aspects of the Local Lemma with applications to routing and partitioning" by the first, third, and fourth authors, and (ii) "A deterministic approximation algorithm for a minmax integer programming problem" by the second author.

†Mathematics Department and the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. Supported in part by Army Grant DAAH04-95-1-0607 and DARPA Grant N00014-95-1-1246. E-mail: `ftl@math.mit.edu`.

‡Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C. Email: `cjlu@iis.sinica.edu.tw`.

§University of California, Berkeley, CA 94705. Part of this work was done while at the NEC Research Institute, 4 Independence Way, Princeton, NJ 08540. E-mail: `satishr@cs.berkeley.edu`.

¶Bell Laboratories, Lucent Technologies, 600–700 Mountain Avenue, Murray Hill, NJ 07974-0636, USA. Parts of this work were done: (i) while at the School of Computing, National University of Singapore, Singapore 119260, supported in part by National University of Singapore Academic Research Fund Grants RP960620 and RP970607; and (ii) while visiting the NEC Research Institute, 4 Independence Way, Princeton, NJ 08540. E-mail: `srin@research.bell-labs.com`.

$G$, how do we construct an $(s_i, t_i)$ path for each $i$, so that the *congestion* (maximum number of paths using any given edge) $C$ is minimized? The multi-commodity flow relaxation of this problem is to send one unit of flow from $s_i$ to $t_i$, for each $i$; the resultant minimum *fractional congestion* (maximum amount of flow using any edge) $y^*$ is a lower bound on $C$. Since this relaxation is, e.g., a linear program (LP), the corresponding optimal multi-commodity flow can be found efficiently. Next suppose we do a flow decomposition: for each $(s_i, t_i)$, we find a set of $\ell_i$ paths $P_{i,j}$ for $j \in [\ell_i]$, with positive flow values that sum to 1. We can assume w.l.o.g. that $\sum_{i\in[k]} \ell_i \leq m$, as each time we can select a path and assign it a flow that saturates at least one edge on it (see, e.g., [21]). Finally, we want to choose exactly one path for each pair to minimize the congestion $C$. The first two steps can be done efficiently, so the final step is our focus.

How large can $C$ be as a function of $y^*$ and of some other parameters of $G$? (Even the special case $y^* = \mathrm{O}(1)$ is of much interest; $C \leq \mathrm{O}(\log|V|/\log\log|V|)$ here [21]. This special case is related to several conjectures and results [23, 15, 17, 14].) Our first family of results, described in Section 1.1, yield new constructive results for generalizations of this question. Our second family of results, discussed in Section 1.2, are related to the question: how large can $y^* < 1$ be if we *require* that $C = 1$, i.e., if we need an edge-disjoint routing?

**1.1. Routing, partitioning, and minmax integer programs.** Our first family of results concern *minmax integer programs* (MIP) as defined in the following; these were named *minimax integer programs* in [25].

DEFINITION 1.1. *An MIP has variables $Y$ and $\{x_{i,j} : i \in [k], j \in [\ell_i]\}$, for some integers $\{\ell_i\}$. Let $N = \sum_{i\in[k]} \ell_i$ and let $x$ denote the $N$-dimensional vector of the variables $x_{i,j}$ (arranged in any fixed order). An MIP seeks to minimize a real $Y$ subject to: (i) $\forall i \in [k]$ $\sum_{j\in[\ell_i]} x_{i,j} = 1$; (ii) a system of linear inequalities $Ax \leq \vec{Y}$, where $A \in [0,1]^{m\times N}$ and $\vec{Y}$ is the $m$-dimensional vector with the variable $Y$ in each component, and (iii) $\forall i, j,\ x_{i,j} \in \{0, 1\}$.*

For convenience, we will call (i) the equality constraints, (ii) the congestion constraints, and (iii) the integrality constraints.

Note that the final step of the low-congestion routing problem can be captured by this form of MIP: the binary variable $x_{i,j}$, for $i \in [k]$ and $j \in [\ell_i]$, indicates whether or not the path $P_{i,j}$ is chosen, and $A_{r,(i,j)}$, for $r \in [m]$, $i \in [k]$, and $j \in [\ell_i]$, indicates whether the edge $r$ is in the path $P_{i,j}$. The MIP problem also arises from a canonical NP-hard hypergraph-partitioning problem [9], which can be applied to support some divide and conquer approaches [2, 12, 16]. We are given a set-system $H = (V, F)$, where $V = [k]$ and $F = \{S_1, \ldots, S_M\} \subseteq 2^V$. Given a positive integer $\ell$, the problem is to partition $V$ into $\ell$ parts, so that *each* $S_j$ is "split well": we want a $\chi : V \to [\ell]$ that minimizes $Y = \max_{j\in[M],q\in[\ell]} |\{i \in S_j :\ \chi(i) = q\}|$. This is easily written as an MIP with $m = M\ell$.

In MIPs, two parameters, $d$ and $\kappa$, are of particular importance to us. Let $d$ denote the maximum number of nonzero entries in any column of the matrix $A$. In the context of the low-congestion routing problem, $d$ is the length of the longest path among the paths $P_{i,j}$. Let $\kappa \geq 1$ be such that the minimum nonzero entry of $A$ is $\kappa^{-1}$. As was for the low-congestion routing problem, an LP relaxation for an MIP is to remove the integrality constraints (iii). The resulting LP optimum $y^*$ is a lower bound on $Y$. As MIPs are NP-hard, we seek approximation algorithms for them: we will present an efficient way of rounding an optimal solution to the LP. The currentbest upper-bounds on $Y$ (as a function of $y^*$ and other parameters) are due to [25], improving on earlier work of [21]. In particular, $Y$ has been shown to be "not much above" $y^*$ if $d$ is "small", i.e., if an MIP instance is *column-sparse*. However, these techniques are non-constructive and are based on an extension of the LLL. Here, we make progress on the algorithmic aspect, i.e., we develop approximation algorithms for MIPs, by focusing on MIP instances where $\kappa$ is not "too large": indeed, note that we have $\kappa = 1$ for MIPs with $\{0, 1\}$ matrices $A$ (e.g., for low-congestion routing and hypergraph-partitioning).

The previous-best result for general MIPs was non-constructive, and showed that $Y \leq y^*(1 + \mathrm{O}(L'(y^*, 1/(2d)))) + \mathrm{O}(1)$ [25], where $L'$ is a function to be defined in Section 2. Here, we present a deterministic polynomial time algorithm that will produce an integral solution of value at most $y^*(1+\mathrm{O}(L'(y^*, 1/(d\kappa))))+\mathrm{O}(1)$. $L'(\mu, p)$ depends logarithmically on $p^{-1}$; thus, for families of MIPs where, e.g., $\kappa$ grows at most as fast as poly$(d)$, we match the previous-best results *constructively*, to within a constant factor. Also, for any family of MIPs where $y^* = \omega(\log(d+\kappa))$ as $d\kappa \to \infty$, our algorithm will produce an integral solution of value $y^*(1 + o(1))$ (e.g., in the hypergraph-partitioning problem, we can asymptotically constructively match the LP optimum if $\ell = o((\max_i |S_i|)/\log d)$, where $d$ is the degree of $H$). For general MIPs, our improvement over the previous-best *constructive* result [21] is in essentially replacing $m$ by $d\kappa$ in the integral solution produced: we get good improvements for MIP instances where $d\kappa \ll m$.

See [6] for recent related results on constructive results for families of MIPs.

**1.2. Constructive sufficient conditions for disjoint paths.** Suppose we require congestion $C = 1$ in the routing problem described before. Let $G = (V, E)$ with $|V| = n$ and $|E| = m$. Randomized rounding approaches show that if $y^* \leq a_0/\sqrt{m}$, then $C = 1$ is achievable constructively; on the other hand, there are instances where $y^* = \Theta(1/\sqrt{n})$ and $C \geq 2$ [11]. Thus, the gap is large in general. An interesting alternative parametrization is based on $d$, the length of a longest path in a fractional optimal solution; the existentially optimal result that $C = 1$ if $y^* \leq a_1/d$, has been shown via the LLL [14]. ($a_0, a_1, a_2$ etc. denote positive constants.) It would be of much interest to make this constructive. As mentioned above, the approaches of [4, 1, 18] does not seem to extend here. We get around this by an approach that generates a large number of paths according to a carefully chosen distribution. Though many of these paths could intersect, we show that if $y^* \leq a_2/(d \ln k)$, then we can locate an edge-disjoint collection within these paths with high probability (whp).

Thus, we present new algorithmic approaches to some classes of packing problems, where the previous-best results were nonconstructive, based on versions of the LLL.

**2. Preliminaries.** Let $X_1, X_2, \ldots, X_n$ be a sequence of independent random variables taking values from $[0, 1]$ with $\mathrm{E}[X_i] = p_i$. Let $X \doteq \sum_{i\in[n]} X_i$ and $\mu \doteq \mathrm{E}[X] = \sum_{i\in[k]} p_i$. Let $S_b$ be the $b$th elementary symmetric polynomial; i.e.,

$$S_b(p_1, \ldots, p_n) \doteq \sum_{I\subseteq[k],|I|=b} \prod_{i\in I} p_i.$$

We summarize the large deviation bounds of Chernoff [5], Hoeffding [10], and Schmidt, Siegel, and Srinivasan [22] in the following:

LEMMA 2.1. *For* $0 \leq \delta \leq 1$, $\Pr[|X - \mu| \geq \mu\delta] \leq 2e^{-\mu\delta^2/3}$.

LEMMA 2.2. *For* $\delta \geq 0$, $\Pr[X \geq \mu(1+\delta)] \leq S_{\lceil\mu\delta\rceil}(p_1, \ldots, p_n)/\binom{\mu(1+\delta)}{\lceil\mu\delta\rceil} \leq H(\mu, \delta)$, *where* $H(\mu, \delta) \doteq (e^\delta/(1 + \delta)^{(1+\delta)})^\mu$.

For $p \in (0,1)$, let $L(\mu, p)$ be the smallest $\delta \geq 0$ such that $H(\mu, \delta) \leq p$. It is not hard to see that

$$L(\mu,p) = \begin{cases} \Theta(\sqrt{\frac{\log p^{-1}}{\mu}}) & \text{if } \mu \geq \frac{1}{2}\log p^{-1}; \\ \Theta(\frac{\log p^{-1}}{\mu \log((\log p^{-1})/\mu)}) & \text{otherwise.} \end{cases}$$

In [25], a related bound $L'(\mu, p)$ was used, which was defined to be the smallest $\delta \geq 0$ such that $\lceil \mu\delta \rceil H(\mu, \delta) \leq p$. So, for this $\delta$, we have

$$\Pr[X \geq \mu(1 + L'(\mu, p))] \leq p/\lceil \mu\delta \rceil.$$

It can also be seen that

$$L'(\mu,p) = \begin{cases} \Theta(\sqrt{\frac{\log(\mu+p^{-1})}{\mu}}) & \text{if } \mu \geq \frac{1}{2}\log p^{-1}; \\ \Theta(\frac{\log p^{-1}}{\mu \log((\log p^{-1})/\mu)}) & \text{otherwise.} \end{cases}$$

**3. Constructive approximations for minmax integer programs.** We now present results for MIPs. Consider a generic MIP and its relaxation, as described in Section 1. Let $\{x^*_{i,j} : i \in [k], j \in [\ell_i]\}$ denote the given optimal solution to the LP relaxation. Starting with this, the algorithm will construct a sequence of feasible solutions to the LP relaxation; variables $x_{i,j}$ that get rounded to 0 or 1 will never have their values re-altered. There are two relevant parameters:

- $\ell$: the maximum, over all $i \in [k]$, of the number of variables $x_{i,j}$ to be rounded[1];
- $t$: the maximum number of variables to be rounded in a row of congestion constraints.

We will show two ways to do the rounding. The first one has a slightly inferior performance, and gives the following theorem.

THEOREM 3.1. *There exists a deterministic polynomial time algorithm for finding a solution of value at most $y^*(1 + O(L(y^*, 1/(dt\ell))))$ for any given MIP.*

The dependency on $t\ell$ in Theorem 3.1 is undesirable, as $t\ell$ sometimes could be large. Next, we show how to do the rounding in several iterations, so that $t$ and $\ell$ decrease in each iteration and finally reach poly$(\kappa, d, y^*)$, with only a slight increase in $y^*$. Then we can apply Theorem 3.1 to get the following theorem. Note that $L(y^*, 1/\text{poly}(\kappa, d, y^*)) = O(L'(y^*, 1/(d\kappa)))$.

THEOREM 3.2. *There exists a deterministic polynomial time algorithm for finding a solution of value $y^*(1 + O(L'(y^*, 1/(d\kappa)))) + O(1)$ for any given MIP.*

Next, we prove the above two theorems.

**3.1. Proof of Theorem 3.1.** Independently for each $i \in [k]$, randomly round exactly one $x^*_{i,j}$ to 1, with $x^*_{i,j}$ chosen with probability $x^*_{i,j}$, and let $z_{i,j}$ be the random variable with the rounded value. Let $z$ be the $N$-dimensional vector of the $z_{i,j}$'s. We say that a row $r \in [m]$ of the congestion constraints is *affected* by another row $r' \in [m]$ if the random variables $(Az)_r$ and $(Az)_{r'}$ are dependent, which occurs exactly when

$$\exists i \in [k],\ \exists j, j' \in [\ell_i] \text{ s.t. } A_{r,(i,j)}, A_{r',(i,j')} \neq 0, \text{ and } x^*_{i,j}, x^*_{i,j'} \notin \{0,1\}.$$

[1]We can assume w.l.o.g. that the LP optimal solution is a *basic* feasible solution, so $\ell \leq m$ and $t \leq m$.

So each row can be affected by at most $D \doteq dt\ell$ other rows. For $r \in [m]$ and $i \in [k]$, define

$$Z_{r,i} \doteq \sum_{j \in [\ell_i]} A_{r,(i,j)} z_{i,j}.$$

Clearly, for each $r \in [m]$, $Z_{r,1}, \ldots, Z_{r,k}$ are a sequence of independent random variables taking values from $[0,1]$, with $\mathrm{E}[\sum_{i \in [k]} Z_{r,i}] = \mathrm{E}[(Az)_r] \leq y^*$; so

$$\Pr[\sum_{i \in [k]} Z_{r,i} > y^*(1 + L(y^*, \frac{1}{4D}))] \leq \frac{1}{4D}.$$

By the LLL, there exists a rounding such that for every $r \in [m]$, $(Az)_r \leq y^*(1 + L(y^*, \frac{1}{4D}))$. We will follow the broad approach of [4, 1] to find a rounding such that for every $r \in [m]$,

$$(Az)_r \leq y^*(1 + O(L(y^*, \frac{1}{D}))).$$

A direct application of the ideas of [4, 1] leads to $(Az)_r \leq cy^*(1 + L(y^*, \frac{1}{D}))$ for some constant $c > 1$, which is not sufficient for use in Theorem 3.2. A twist is needed to achieve our bound.

We first show a randomized algorithm, and then derandomize it.

**3.1.1. A Randomized Algorithm.** For a row $r \in [m]$ and for $i \in [k], j \in [\ell_i]$ with $x^*_{i,j} \notin \{0,1\}$, we say that $z_{i,j}$ is a variable of $r$ if $A_{r,(i,j)} > 0$. Consider the graph $G(V, E)$, where $V = [m]$ represents the $m$ rows, and two nodes are adjacent iff one can be affected by the other. Note that each node has at most $D$ neighbors. Let $G^{(a,b)}(V, E')$ be the graph with two nodes adjacent iff their distance is exactly $a$ or $b$ in $G$. Call a set of nodes in $G$ an $(a,b)$-tree if they form a connected component in $G^{(a,b)}$. Call a node $r \in [m]$ *bad* if

$$(Az)_r = \sum_{i \in [k]} Z_{r,i} > y^*(1 + L(y^*, \frac{1}{6D^4})),$$

which happens with probability at most $\frac{1}{6D^4}$. For a (1, 2)-tree $T$ and a node $r \in [m]$, let $I_{r,T}$ denote the set of indices $i \in [k]$ such that $Z_{r,i}$ can be affected by re-rounding variables in the rows represented by $T \setminus \{r\}$; i.e., $I_{r,T}$ is the set

$$\{i \in [k] : \exists r' \in T \setminus \{r\}, \exists j, j' \in [\ell_i] \text{ s.t. } A_{r,(i,j)}, A_{r',(i,j')} \neq 0, \text{ and } x^*_{i,j}, x^*_{i,j'} \notin \{0,1\}\}.$$

Let $\overline{I}_{r,T} \doteq [k] \setminus I_{r,T}$. Call a node $r$ *bad for $T$* if

$$\sum_{i \in \overline{I}_{r,T}} Z_{r,i} > \mathrm{E}[\sum_{i \in \overline{I}_{r,T}} Z_{r,i}] + y^* L(y^*, \frac{1}{6D^4}); \tag{3.1}$$

for any fixed $r$, $r$ is bad for $T$ with probability at most $\frac{1}{6D^4}$, by the definition of $L$. Call a (1, 2)-tree $T$ *bad*, if every node $r \in T$ is either bad or bad for $T$.

The intuition is that our randomized rounding is unlikely to lead to large bad (1, 2)-trees and that variables in different bad (1, 2)-trees can be re-rounded separately. Our algorithm consists of at most three phases. In the first phase, we find a rounding such that all bad (1, 2)-trees have size at most $2D \log m / \log D$. Then we try to re-round variables in each bad (1, 2)-tree separately. If $m$ is small, the re-rounding can be done in the second phase. Otherwise we need another phase.

**Phase 1.** First we need a lemma.

LEMMA 3.3. *The probability that our randomized rounding leads to a bad (1, 2)-tree of size at least* $2D \log m / \log D$ *is at most* $1/m$.

*Proof.* Let $u = 2 \log m / \log D$. As a (1, 2)-tree of size $Du$ must contain a (2, 3)-tree of size $u$ and two nodes of a (2, 3)-tree do not affect each other, a (1, 2)-tree of size $Du$ is bad with probability at most $(\frac{1}{6D^4} + \frac{1}{6D^4})^u = (\frac{1}{3D^4})^u$. The number of (1, 2)-trees of size $Du$ is at most $\frac{m}{(D^2-1)Du+1}\binom{D^3u}{u} < m(\frac{3D^3u}{u})^u = m(3D^3)^u$: see Appendix A. So the probability that we get a bad (1, 2)-tree of size $Du$ is at most

$$m(3D^3)^u(\frac{1}{3D^4})^u \leq m(\frac{1}{D})^u \leq \frac{1}{m}.$$

$\square$

This gives us a randomized algorithm for finding a rounding with no large bad (1, 2)-trees. In Section 3.1.2, we will show how to find such a rounding deterministically, by using the method of conditional probabilities [20].

**Phase 2.** Suppose we have found a rounding with no bad (1, 2)-tree of size $Du$. Consider a maximal bad (1, 2)-tree $T$. We will re-round all variables in $T$ using the same randomized approach as in Phase 1. Now the neighbors of $T$ become *dangerous* as they are affected by $T$ and could turn bad after the re-rounding. Let $N(T)$ denote the set of $T$'s neighbors. Note that nodes in $T$ cannot be affected by other bad (1, 2)-trees and nodes in $N(T)$ are not dangerous for other bad (1, 2)-trees. So we can deal with each bad (1, 2)-tree separately.

For $r \in N(T)$, since $T$ is maximal and $r \notin T$, we see from (3.1) that

$$\sum_{i \in \overline{I}_{r,T}} Z_{r,i} \leq \mathrm{E}[\sum_{i \in \overline{I}_{r,T}} Z_{r,i}] + y^* L(y^*, \frac{1}{6D^4}). \tag{3.2}$$

After the re-rounding, we call a node $r \in T \cup N(T)$ bad if

$$(Az)_r > y^*(1 + 2L(y^*, \frac{1}{6D^4})).$$

For $r \in T$, this obviously happens with probability at most $\frac{1}{6D^4}$. For $r \in N(T)$, as $\sum_{i \in \overline{I}_{r,T}} Z_{r,i}$ is not affected, we get from (3.2) that $r$ is bad only if

$$\sum_{i \in I_{r,T}} Z_{r,i} > \mathrm{E}[\sum_{i \in I_{r,T}} Z_{r,i}] + y^* L(y^*, \frac{1}{6D^4}),$$

which happens with probability at most $\frac{1}{6D^4}$.

$T \cup N(T)$ has at most $Du + D^2u = D(D+1)u$ nodes. Suppose $\sqrt{\log m / \log\log m} \leq D$. Then $u = 2\log m / \log D \leq 5 \log m / \log\log m \leq 5D^2$. So the probability of having a bad node is at most

$$D(D+1)u\frac{1}{6D^4} < 5(D+1)D^3\frac{1}{6D^4} < 1.$$

We can find a good re-rounding in deterministic polynomial time via the method of conditional probabilities. The proof is omitted here, as the idea is similar to that in Section 3.1.2.

Otherwise, we have $D < \sqrt{\log m / \log\log m}$. Using a procedure similar to that in phase 1, we can find a re-rounding such that all bad (1, 2)-trees have size at most

$$O(D \log(D^2 \log m)/\log D) = O(\sqrt{\log m \log\log m}/\log D).$$

Then we enter phase 3.

**Phase 3.** The LLL shows that a good re-rounding exists. Since: (i) each bad (1, 2)-tree has at most $\beta = O(\sqrt{\log m \log\log m}/\log D)$ nodes, and (ii) $t \le D \le \sqrt{\log m/\log\log m}$, each bad (1, 2)-tree has at most $t\beta = O(\log m/\log D)$ variables. So, we can use an exhaustive search in deterministic poly($m$) time to find a good re-rounding such that for every $r \in [m]$,

$$(Az)_r \le y^*(1 + 3L(y^*, \frac{1}{6D^4})).$$

**3.1.2. Derandomization of Phase 1.** Let $R$ denote the set of all (1, 2)-trees of size $Du = 2D\log m/\log D$. From Lemma 3.3, a randomized rounding is unlikely to result in some tree in $R$ being bad. How do we find a good rounding deterministically? The idea is to use the standard *method of conditional probabilities* with a pessimistic estimator [20]. For $i \in [k]$, let $x_i^*$ denote the vector $(x_{i,1}^*, \ldots, x_{i,\ell_i}^*)$, and let $z_i$ denote $(z_{i,1}, \ldots, z_{i,\ell_i})$. We want to round $x_i^*$ to $z_i$ one at a time, subject to the equality constraints of the MIP. The value of $z_i$ is chosen to minimize the probability of some tree in $R$ being bad if we randomly round the remaining $x_{i+1}^*, \ldots, x_k^*$, conditional on the previously chosen $z_1, \ldots, z_{i-1}$. The hope is that the final rounding is a good one because the final conditional probability, which is either 0 or 1, is at most the original unconditional one, which is less than 1. However, it is not easy to compute the exact conditional probability at each step. So we use a pessimistic estimator instead.

For $T \in R$, let $T_{23}$ be an arbitrary maximal (2, 3)-tree in $T$. Suppose that we have already fixed $z_1, \ldots, z_i$. Then

$$\begin{aligned} P_i(z_1, \ldots, z_i) &\doteq \Pr_{z_{i+1},\ldots,z_k}[\exists T \in R, T \text{ is bad} \mid z_1, \ldots, z_i] \\ &\le \sum_{T\in R} \prod_{r \in T_{23}} \Pr_{z_{i+1},\ldots,z_k}[r \text{ is bad or bad for } T \mid z_1, \ldots, z_i]. \end{aligned}$$

Let $b \doteq \lceil y^* L(y^*, \frac{1}{6D^4})\rceil$ and $c \doteq y^*(1 + L(y^*, \frac{1}{6D^4}))$. Let $w_{r,T} \doteq \mathrm{E}[\sum_{j\in \overline{I}_{r,T}} Z_{r,j}] + y^* L(y^*, \frac{1}{6D^4})$, and let $S_b^r(I)$ denote $S_b$ on input $(\mathrm{E}[Z_{r,v}] : v \in I)$, for $I \subseteq [k]$. Note that

$$\mathrm{E}[Z_{r,v}] = \begin{cases} \sum_{j\in[\ell_v]} A_{r,(v,j)} z_{v,j} & \text{if } z_v \text{ has been fixed,} \\ \sum_{j\in[\ell_v]} A_{r,(v,j)} x_{v,j}^* & \text{otherwise.} \end{cases}$$

Consider the following pessimistic estimator:

$$A_i(z_1, \ldots, z_i) \doteq \sum_{T\in R} \prod_{r\in T_{23}} \left( \frac{S_b^r([k])}{\binom{c}{b}} + \frac{S_b^r(\overline{I}_{r,T})}{\binom{w_{r,T}}{b}} \right).$$

From Lemma 2.2, we have $P_i(z_1,\dots,z_i) \le A_i(z_1,\dots,z_i)$. Now,

$$\begin{aligned} A_i(z_1,\dots,z_i) &= \sum_{T\in R}\prod_{r\in T_{23}} \mathrm{E}_{z_{i+1}}\left[\frac{S_b^r([k])}{\binom{c}{b}} + \frac{S_b^r(\overline{I}_{r,T})}{\binom{w_{r,T}}{b}}\right] \\ &= \sum_{T\in R} \mathrm{E}_{z_{i+1}}\left[\prod_{r\in T_{23}}\left(\frac{S_b^r([k])}{\binom{c}{b}} + \frac{S_b^r(\overline{I}_{r,T})}{\binom{w_{r,T}}{b}}\right)\right] \\ &= \mathrm{E}_{z_{i+1}}[A_{i+1}(z_1,\dots,z_{i+1})], \end{aligned}$$

where the second equality is because two distinct nodes of $T_{23}$ share no variable. We choose $z_{i+1}$ to minimize $A_{i+1}(z_1,\dots,z_{i+1})$. Then we have

$$\frac{1}{m} \ge A_0 \ge A_1(z_1) \ge \cdots \ge A_k(z_1,\dots,z_k) \ge P_k(z_1,\dots,z_k).$$

$P_k(z_1,\dots,z_k)$ is either 1 or 0 depending on whether or not the rounding results in a bad (2, 3)-tree of size $u$. As $P_k(z_1,\dots,z_k) < 1$, we have found a rounding such that there is no bad (1, 2)-tree of size $Du$.

It remains to show that for any $i$ and $z_1,\dots,z_i$, $A_i(z_1,\dots,z_i)$ can be computed efficiently. There are poly$(m)$ number of (1, 2)-trees of size $Du$ in $R$, and they can be enumerated in polynomial time (see Appendix A). Next, we show how to compute $S_b$ efficiently. For any $y_1,\dots,y_k$, note that

$$S_b(y_1,\dots,y_k) = y_k \cdot S_{b-1}(y_1,\dots,y_{k-1}) + S_b(y_1,\dots,y_{k-1}).$$

Thus, using a dynamic programming approach that is similar to the standard method of computing Pascal's triangle, we can compute $S_b$ in deterministic polynomial time.

So $A_i$ can be computed in deterministic polynomial time.

**3.2. Proof of Theorem 3.2.** Let $x = x^*$, $y = y^*$, and $D = dt\ell$. Note that we can just apply Theorem 3.1 if the following holds:

$$D \le \kappa^8 d^8 \text{ or } D \le y^8 \text{ or } D \le c_0 \text{ or } D \le y^{-8/5}. \tag{3.3}$$

Here, $c_0$ is some absolute constant to be chosen after equation (3.6). So suppose (3.3) is not true; we will try to reduce $D$ in several iterations to finally reach this condition. The idea is similar to that in [25], and we make the key observation that the approach of [25] can reduce not only $\ell$ but also $t$. So we can use the standard version of the LLL, instead of the generalization of the LLL developed in [25]. We consider two cases depending on the range of $y$.

**Case 1:** $y \ge 1$. We will scale up each $x_{i,j}$, do a rounding, and then normalize each $x_{i,j}$ back to satisfy the equality constraint. This is done as follows.

Recall the notion of randomized rounding of a real [21]: given a real $\alpha \ge 0$, we round it to $\lceil\alpha\rceil$ with probability $\alpha - \lfloor\alpha\rfloor$, and round it to $\lfloor\alpha\rfloor$ with probability $1-(\alpha-\lfloor\alpha\rfloor)$. Let $x'_{i,j} = x_{i,j}y^2\log^3 D$. We will randomly round each $x'_{i,j}$ *independently* (notice the difference from the rounding in Theorem 3.1). Let $z_{i,j}$ be the random variable taking the value $\lceil x'_{i,j}\rceil$ with probability $x'_{i,j} - \lfloor x'_{i,j}\rfloor$, and taking the value $\lfloor x'_{i,j}\rfloor$ with probability $\lceil x'_{i,j}\rceil - x'_{i,j}$. Note that $\mathrm{E}[z_{i,j}] = x'_{i,j}$; hence, for each $r \in [m]$, we have $\mathrm{E}[(Az)_r] \le y^3\log^3 D$. Let $R_r$ denote the bad event that

$$(Az)_r > y^3\log^3 D(1 + L(y^3\log^3 D, \frac{1}{6D^4})),$$

with $\Pr[R_r] \le \frac{1}{6D^4}$. We will call events such as $R_r$ as "type $R$" events. For each $i \in [k]$, we have $\mathrm{E}[\sum_{j\in[\ell_i]} z_{i,j}] = y^2 \log^3 D$; let $S_i$ denote the bad event that

$$|\sum_{j\in[\ell_i]} z_{i,j} - y^2 \log^3 D| > 4y \log^2 D,$$

with $\Pr[S_i] \le \frac{1}{6D^4}$ from Lemma 2.1. Call these "type $S$" events.

Each type $R$ event depends on at most $dt$ type $R$ events and at most $t$ type $S$ events. Each type $S$ event depends on at most $d\ell$ type $R$ events and on no other type $S$ events. There are $m' = m + k$ bad events, each happening with probability at most $\frac{1}{6D^4}$, and each depending on at most $D$ other bad events. From the LLL, there exists a rounding such that no bad event happens. An algorithm similar to that in the previous section can be used to find a good rounding in deterministic polynomial time, as sketched below. In phase 1, we find a rounding such that all bad (1, 2)-trees have size at most $2D \log m' / \log D$. In phase 2, either we can find a good re-rounding and we are done, or we can find a re-rounding such that all bad (1, 2)-trees are very small. Then we can use an exhaustive search to find a good re-rounding in phase 3. After this, we have that for all $r \in [m]$,

$$(Az)_r \le y^3 \log^3 D(1 + O(1/(y^{1.5} \log D)))$$

and for all $i \in [k]$,

$$|\sum_{j\in[\ell_i]} z_{i,j} - y^2 \log^3 D| \le O(y \log^2 D).$$

Then we do the normalization

$$x_{i,j} = \frac{z_{i,j}}{\sum_{v\in[\ell_i]} z_{i,v}},$$

to satisfy the equality constraints. Now for every $r \in [m]$,

$$(Ax)_r \le \frac{y^3 \log^3 D(1 + O(1/(y^{1.5} \log D)))}{y^2 \log^3 D(1 - O(1/(y \log D)))} = y(1 + O(\frac{1}{y \log D})); \tag{3.4}$$

so, the values $(Ax)_r$ increase just a little. Notice that each $z_{i,j}$ is a non-negative integer. So for each $i \in [k]$, the number of nonzero $z_{i,j}$ (or $x_{i,j}$), for $j \in [\ell_i]$, is at most

$$\sum_{j\in[\ell_i]} z_{i,j} = O(y^2 \log^3 D) = O(D^{1/4} \log^3 D), \tag{3.5}$$

since (3.3) is not true. (The constants hidden in the "big-Oh" notation in (3.4) and in (3.5), as well as in the remaining big-Oh upper bounds in this subsection, are all absolute constants that are *independent* of $c_0$.) Next, recall that every nonzero entry of $A$ is at least $\kappa^{-1}$. So, for each $r \in [m]$, the number of nonzero $A_{r,(i,j)} x_{i,j}$ is at most

$$\kappa(Az)_r = O(\kappa y^3 \log^3 D) = O(\kappa D^{3/8} \log^3 D).$$

The parameters $\ell$ and $t$ drop to $O(D^{1/4} \log^3 D)$ and $O(\kappa D^{3/8} \log^3 D)$ respectively. Thus there is an absolute constant $c_0'$, independent of $c_0$, such that the parameter $D$ comes down to

$$\begin{aligned} c_0' d\kappa D^{5/8} \log^6 D &\le c_0' \kappa^{-1} D^{1/8} \kappa D^{5/8} \log^6 D \\ &= c_0' D^{3/4} \log^6 D, \end{aligned} \tag{3.6}$$

which is at most $D^{7/8}$ if we choose $c_0$ large enough such that for all $D > c_0$, the bound $c_0' \log^6 D < D^{1/8}$ holds. Then repeating the above process $O(\log\log D)$ times will ensure that $D$ becomes small enough to satisfy condition (3.3).

Finally, how large can $y$ become? Suppose $y_0, y_1, \ldots, y_s$ are the successive values assumed by $y$, and $D_0, D_1, \ldots, D_s$ are the successive values assumed by $D$, in the above process. We see from (3.6) that if $c_0$ is large enough, then $\log D_{i+1} \leq (7/8)\log D_i$; hence, $\log D_i \geq (8/7)^{i-s}\log D_s$. Thus we see from (3.4) that

$$y_{i+1} \leq y_i + O(1/\log D_i) \leq y_i + O((7/8)^{s-i}(\log D_s)^{-1});$$

a telescoping sum shows that $y_s \leq y_0 + O(1)$. Now that (3.3) holds and $y$ has increased by at most a constant, we can apply Theorem 3.1.

**Case 2:** $y < 1$. The idea is the same here. The scaling up is now $x'_{i,j} = x_{i,j} y^{-1} \log^3 D$, for each $i \in [k]$ and $j \in [\ell_i]$, and the rounding and normalization are similar.

**4. Constructive sufficient conditions for disjoint paths.** We now move on to the problem of lower bounds on $y^*$ in our low-congestion routing problem that can guarantee algorithmically that the integral congestion $C$ is 1. Our main result here is Theorem 4.1. In typical applications of it, the parameter $p$ is likely to be $o(1)$ or bounded away from 1. So, the term "$\lceil \ln p^{-1} \rceil$" can be taken to be "$\ln p^{-1}$" for a first reading.

As will be seen below, we will employ certain randomized rounding and "scaling up–randomized rounding–scaling down" ideas similar to those of Section 3.2. Also, one of our original applications of Theorem 4.1 was for edge-disjoint paths in *expander graphs*; however, this problem has been completely resolved by Frieze recently [8]. In Section 4.1, we discuss why approaches such as those of [4, 1, 18] do not appear to yield Theorem 4.1.

THEOREM 4.1. *There is a constant $c_1 > 0$ such that the following holds. Suppose we are given any (directed or undirected) graph $G$, any $k$ (multi-)sets of paths in it, $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$, and any parameter $p \in (0,1)$. Let the paths in $\mathcal{P}_i$ be denoted $P_{i,j}$; let $d$ be the maximum, over all $i, j$, of the length of $P_{i,j}$. Let each $P_{i,j}$ carry some non-negative amount of flow such that: (i) the total flow in each $\mathcal{P}_i$ is 1, and (ii) for each edge in $G$, the total flow using it is at most $c_1/(d\lceil \ln p^{-1}\rceil)$. Then, there is a randomized polynomial algorithm to pick one path each from at least $k(1-p)$ of the $\mathcal{P}_i$, such that all paths chosen are edge-disjoint whp. (If we want one path from each of the $\mathcal{P}_i$, we can set, e.g., $p = 1/(2k)$.)*

Specializing to the case where all paths in any given $\mathcal{P}_i$ connect the same pair of vertices, we get

COROLLARY 4.2. *There is a constant $c_1 > 0$ such that the following holds. Suppose we are given any (directed or undirected) graph $G$, any collection $\mathcal{T} = \{(s_i, t_i) : i \in [k]\}$ of vertex-pairs in $G$, and any parameter $p \in (0,1)$. Suppose there is a unit flow from $s_i$ to $t_i$ for each $i$, such that the fractional congestion is at most $c_1/(d\lceil \ln p^{-1}\rceil)$; $d$ denotes the length of a longest flow-path. Then, there is a randomized polynomial algorithm to pick an $(s_i, t_i)$-path for at least $k(1-p)$ indices $i$, such that all paths chosen are edge-disjoint whp. (If we want one $(s_i, t_i)$-path for each $i$, we can set $p = 1/(2k)$.)*

**Remark.** Our algorithm will be easily seen to run in polynomial time, and will involve several random processes. For each process for which success or failure can be checked efficiently, we will just show that the process succeeds with constant probability. This suffices since, letting $N$ denote the "input size" for the problem, we can repeat each such process, say $O(N)$ times, to drive the failure probability down to $2^{-\Theta(N)}$. Thus, our algorithm will succeed whp.

We now present the algorithm and proof for Theorem 4.1.

*Proof.* We will choose the constant $c_1 \in (0,1)$ sufficiently small. Let $z \doteq d\lceil \ln p^{-1} \rceil / c_1$; thus, the *fractional congestion* (maximum flow using any given edge) is at most $1/z$. There are three simple steps in the algorithm:

**Step 1.** This is a "scaling up–randomized rounding–scaling down" step, similar to the one employed in Cases 1 and 3 of Section 3.2. Let $n$ be the number of vertices in $G$. Recall the notion of randomized rounding of a real $\alpha$ [21] from Case 1 of Section 3.2. For a suitably large absolute constant $c_2 > 2$ (say 20), we scale all the flow values by $c_2 z \ln n$, and then conduct randomized rounding independently on each flow value (i.e., for each path $P_{i,j}$). By a standard analysis via Lemmas 2.1 and 2.2, we have whp that:

**(P1)** the total flow $f_i$ in each $\mathcal{P}_i$ is such that $c_2 z(\ln n)/2 \le f_i \le 2c_2 z \ln n$, **and**
**(P2)** the total flow on any edge is at most $2c_2 \ln n$.

(This can also be derandomized easily by the method of conditional probabilities.) Dividing the flow value on each $P_{i,j}$ by $f_i$, we restore the property that the total flow value in each $\mathcal{P}_i$ is 1. Furthermore, the fractional congestion now is at most

$$\begin{aligned} 2c_2 \ln n (\min_i \ f_i)^{-1} &\le 2c_2 \ln n (c_2 z(\ln n)/2)^{-1} \\ &= 4/z. \end{aligned} \tag{4.1}$$

Let $x \doteq \max_i f_i$; (P1) shows that

$$c_2 z(\ln n)/2 \le x \le 2c_2 z \ln n. \tag{4.2}$$

Since the randomized rounding leads to integral flow values, we see, for any $i$, that the number of flow-paths $P_{i,j}$ carrying nonzero flow now is at most $f_i$; hence, the maximum number of flow-paths $P_{i,j}$ carrying nonzero flow now, for any $i$, is at most $x$.

**Step 2.** Suppose, after the scaling-rounding-rescaling process of Step 1, that flow path $P_{i,j}$ carries a flow of value $g_{i,j} \ge 0$. For each $(i,j)$, we proceed as follows. Let $g'_{i,j}$ be the largest multiple of $1/(2x)$ that is no larger than $g_{i,j}$; replace $P_{i,j}$ by $2xg'_{i,j}$ copies of itself, each of the copies carrying a flow of precisely $1/(2x)$. We refer to the new paths as $P'_{i,\ell}$. Fix $i$. How many flow paths do we have in any $\mathcal{P}_i$ now? Since each flow value is now exactly $1/(2x)$ and the total flow in $\mathcal{P}_i$ is now at most 1, this number can be at most $2x$. On the other hand, recall that $\sum_j g_{i,j} = 1$, and, as seen above, that there are at most $x$ (nonzero-flow) paths $P_{i,j}$. Since $g'_{i,j} \ge g_{i,j} - 1/(2x)$, we see by summing that $\sum_j g'_{i,j} \ge 1 - x/(2x) = 1/2$. Thus, since each of the new flow-paths carries a flow of exactly $1/(2x)$, there must be at least $x$ many $P'_{i,j}$. So, for each $i$, the set $A_i$ of flow-paths $P'_{i,j}$ is such that

$$x \le |A_i| \le 2x. \tag{4.3}$$

In the rest of the algorithm description, "flow paths" shall refer to these newly constructed flow paths $P'_{i,j}$.

Let $y$ denote the maximum number of flow-paths $P'_{i,j}$ that use any given edge. Now, all paths $P'_{r,s}$ carry a flow of exactly $1/(2x)$. Thus, by (4.1),

$$y \le (4/z)/(1/(2x)) = 8x/z \le 8(2c_2 z \ln n)/z = 16c_2 \ln n, \tag{4.4}$$

the second inequality following from (4.2).

**Step 3.** Let $t \doteq \lceil \ln p^{-1} \rceil$. Independently for each $i \in [k]$, we do the following. For each $i$, we randomly select $t$ flow-paths (without replacement) from among the collection of flow-paths $P'_{i,j}$. (The bounds $c_2 > 2$, (4.3), and (4.2) show that $|A_i| \geq z \ln n$. Now, $t = c_1 z/d \leq z \ln n$ since $c_1 < 1$. So our random-selection process is well-defined.) Call all the paths selected here *tentatively chosen*. We will say that two paths $P'_{a,b}$ and $P'_{r,s}$ are *neighbors* iff $a \neq r$ **and** $P'_{a,b}$ and $P'_{r,s}$ share a common edge.

Call $P'_{i,j}$ *bad* iff it and some neighbor of it, are both tentatively chosen; call $i$ dangerous iff all its $t$ tentatively chosen paths $P'_{i,j}$ are bad. We will next prove Theorem 4.3; let us see why this will help us complete the proof of Theorem 4.1. As usual, the probability of 2/3 in Theorem 4.3 can be boosted by repeating the random process sufficiently many times. Thus, we will be able to efficiently: (i) select some $T \subseteq [k]$ of cardinality at least $k(1-p)$, and (ii) pick one tentatively chosen $P'_{i,j}$ for each $i \in T$, such that all the paths picked are edge-disjoint. □

THEOREM 4.3. *If $c_1 > 0$ is sufficiently small, then with probability at least 2/3, the number of dangerous $i$ is at most $kp$.*

*Proof.* Since each path passes through at most $d$ edges and since each edge has at most $y$ flow-paths using it (by definition of $y$), we see that each flow-path has at most $dy$ neighbors. Fix $i$. We now bound the probability that $i$ is dangerous. Suppose $i$ is dangerous, and that it has a collection $S$ of $t$ bad paths $P'_{i,j}$. For each path in $S$, choose one among its bad neighbors arbitrarily. Suppose there are $q$ such **distinct** bad neighbors $p_1, p_2, \ldots, p_q$ of the elements of $S$, where $1 \leq q \leq t$. We can represent this by an unordered $q$-tuple $\langle [S_1, p_1], [S_2, p_2], \ldots, [S_q, p_q] \rangle$, where: (i) the sets $S_j$ form a partition of $S$, and (ii) $p_j$ is a bad neighbor of all elements of $S_j$. Consider any such fixed tuple $T = \langle [S_1, p_1], [S_2, p_2], \ldots, [S_q, p_q] \rangle$. Suppose, for some distinct indices $u_1, u_2, \ldots, u_r$, that the $t+q$ paths in $T$ are: $\ell_1$ paths from $A_{u_1}$, $\ell_2$ paths from $A_{u_2}$, $\ldots$, $\ell_r$ paths from $A_{u_r}$. (The positive integers $r$ and $\ell_1, \ell_2, \ldots, \ell_r$ are such that $\ell_j \leq t$ and $\sum_j \ell_j = t + q$.) The probability of all of $T$'s $(t+q)$ paths getting tentatively chosen is

$$(4.5)\quad \left( \prod_{j=1}^{r} \left[ \binom{|A_{u_j}| - \ell_j}{t - \ell_j} \Big/ \binom{|A_{u_j}|}{t} \right] \right) \leq \prod_{j=1}^{r} (t/|A_{u_j}|)^{\ell_j} \leq \prod_{j=1}^{r} (t/x)^{\ell_j} = (t/x)^{t+q};$$

the second inequality is a consequence of (4.3).

Fix $q$. We next upper-bound the number of possible unordered $q$-tuples $\langle\, [S_1, p_1], [S_2, p_2], \ldots, [S_q, p_q] \,\rangle$. To do so, we will bound the number of such *ordered* $q$-tuples, and divide the result by $q!$. Let us now bound the number of ordered $q$-tuples. Clearly, we may assume that each $S_j$ is nonempty. We first fix the cardinalities of the $S_j$: this can be done in $\binom{t-1}{q-1}$ ways, since $\sum_j |S_j| = t$ and as each $S_j$ is nonempty. Suppose $|S_j| = t_j$. Recall from (4.3) that a member $P'_{i,\ell}$ of $S_1$ can be picked in at most $2x$ ways. Given this choice, since $p_1$ is a neighbor of $P'_{i,\ell}$, $p_1$ can be picked in at most $dy$ ways. Next, since the remaining $t_1 - 1$ elements of $S_1$ are all neighbors of $p_1$, they can be picked in at most $\binom{dy}{t_1-1}$ ways. Thus, the number of ways of choosing $[S_1, p_1]$ is at most

$$(4.6)\qquad (2x) \cdot (dy) \cdot \binom{dy}{t_1 - 1} \leq (2x) \cdot (dy)^{t_1}.$$

Arguing similarly for $[S_2, p_2], \ldots, [S_q, p_q]$, we see, for each fixed choice of $t_1, t_2, \ldots, t_q$, that the number of ordered $q$-tuples is at most $\prod_{j=1}^{q} ((2x) \cdot (dy)^{t_j}) = (2x)^q \cdot (dy)^t \leq$

$2^t x^q \cdot (dy)^t$. Next, as seen above, (i) there are

$$\binom{t-1}{q-1} \le \binom{t}{q} \le e^q t^q / q^q \le e^t t^q / q^q$$

ways of choosing the $t_i$; (ii) the number of unordered $q$-tuples is the number of ordered $q$-tuples divided by $q!$. Combining with (4.5), the probability of existence of some $q$-tuple is at most

$$c_3^t x^q (dy)^t (t/q)^q (t/x)^{t+q}/q! \le c_4^t (t/q)^{2q} (tdy/x)^t, \tag{4.7}$$

where $c_3, c_4 > 0$ are absolute constants; in going from the l.h.s. to the r.h.s., we have used the fact that $q! \ge (q/e)^q \ge q^q/e^t$. By calculus, it is evident that $(t/q)^{2q} \le e^{2t/e}$. Next, by recalling that $z = dt/c_1$ and by using the lower bound on $x$ and upper bound on $y$ from (4.2) and (4.4), we see from (4.7) that the probability of existence of some $q$-tuple is at most $(c_5 c_1)^t$, where $c_5$ is an absolute constant which, in particular, is independent of $c_1$. Finally, since $q$ can only take $t$ values–from 1 to $t$–the probability that $i$ is dangerous is at most $t(c_5 c_1)^t$, which can be made, say, at most $p/3$ by taking $c_1$ appropriately small (recall that $t = \lceil \ln p^{-1} \rceil$). Thus, by Markov's inequality, the probability that there are more than $kp$ dangerous $i$ is at most $1/3$. □

We now sketch two ways in which the constant factor $c_1$ in Theorem 4.1 can be increased; however, these methods do not replace $c_1$ by any super-constant term. First, in (4.6), we can upper-bound $\binom{dy}{t_1-1}$ by $(dy)^{t_1-1}/(t_1-1)!$. More interestingly, we can include an additional element of randomness in Step 3 of our algorithm as follows, motivated by a "random permutation" idea of [24]. Choose a random permutation $\sigma$ of all the paths $P_{i,j'}$, and call $P_{i,j'}$ bad iff it and some neighbor of it *that precedes* $P_{i,j'}$ *in* $\sigma$, are both tentatively chosen. Call $i$ dangerous iff all its $t$ tentatively chosen paths $P'_{i,j}$ are bad, and note that we can efficiently select one tentatively chosen path for each non-dangerous index $i$, in such a way that these selected paths are edge-disjoint. This idea helps improve the constant $c_1$.

**4.1. Comparison of Theorem 4.1 with previous methods.** We now give a brief sketch of an approach of [18], which generalizes the basic algorithm of [4, 1] to cover many new applications. The reader is referred to these three beautiful papers for more details. Suppose we have a set $\mathcal{F} = \{f_1, f_2, \ldots, f_t\}$ of *independent* random variables, each $f_j$ taking values in some domain of cardinality at most $\gamma$. Suppose we also have ("bad") events $E_1, E_2, \ldots, E_m$, such that each $E_i$ is determined by the values of the variables in some $F_i \subseteq \mathcal{F}$. Define $u \doteq \max_i |F_i|$. Let us say that $E_i$ *intersects* $E_j$ iff $F_i \cap F_j \neq \emptyset$; we let $D$ denote the maximum number of other $E_j$ that any given $E_i$ intersects. Let $q = \max_i \Pr[E_i]$. One of the main results of [18] is that if

$$qD^9 < 1/8, \tag{4.8}$$

then we can find an assignment for all the $f_i$ in randomized poly$(m, t, \gamma^{uD \log\log m})$ time such that $\bigwedge_{i \in [m]} \overline{E_i}$ holds.

We now sketch this algorithm. A setting $\mathcal{A}$ of some of the random variables $f_j$ is done in a way such that $\Pr[E_i | \mathcal{A}] \le q^{2/3}$ for all $i$. More importantly, as seen in Section 3.1.1, the problem of setting the remaining $f_j$ gets split into different "components", where each component has at most $D \log m$ events $E_i$ whp; we can handle each component separately. If need be, we repeat the above process, and end up whp with components of size $\mathrm{O}(D \log(D \log m))$; the probability of each $E_i$ conditional on the $f_j$ fixed so far, will be at most $q^{1/3}$. The LLL and (4.8) show that there is a setting of the variables in each component such that all the $E_i$ are avoided. Since each component is quite small now, we can now conduct an exhaustive search separately in each component. In particular, if $\gamma$ and $u$ are "small", say bounded by a constant which is true for many applications of the LLL, the condition (4.8) that is somewhat stronger than the "$eq(D+1) \leq 1$" of the LLL, ensures a constructive version of the LLL. However, as pointed out in [18], if $\gamma$ is "large", say $(t+m)^{\Theta(1)}$, this method does not guarantee polynomial-time algorithms.

Suppose we were to follow the methodology of [4, 1, 18] to try and prove Theorem 4.1. Motivated by the notation above, let us define $\gamma$ to be the maximum number of paths in any (multi-)set $\mathcal{P}_i$. In Theorem 4.1, consider the case where we wish to choose one path from each $\mathcal{P}_i$; i.e., $p = \Theta(1/k)$. Combined with the work of [14], the approach sketched above would require that the fractional congestion in Theorem 4.1 be $\mathrm{O}(1/d\gamma^c)$, for a certain constant $c > 0$; some basic features of the approaches of [4, 1, 18] make it potentially impossible to eliminate this constant. Thus, situations where $\gamma$ is "large" pose problems if we use the approach of [4, 1, 18]. Via the approach of Step 1 of our algorithm for Theorem 4.1, we can reduce $\gamma$ to $\mathrm{O}(d \log n)$, but reducing $\gamma$ to $o(d)$, say, is a challenging open problem. Also, as pointed out in [18] and above, situations where $\gamma$ is "large", say $n^{\Theta(1)}$, do not necessarily admit of polynomial-time solutions via the approach of [4, 1, 18]. Thus, for instances where $\min\{\gamma, d\}$ is, say, $n^{\Theta(1)}$, it is not clear how to use the methods of [4, 1, 18] to derive polynomial-time algorithms.

**5. Open Questions.** The following problems suggest themselves. Can we get a full algorithmic version of the result of [25]? That is, can we get rid of the "$\kappa$" term in Theorem 3.2? Another question is whether $L'$ can be replaced by $L$ in Theorem 3.2. Finally, in the notation of Section 1.2, Corollary 4.2 shows that "$y^* \leq a_2/(d \ln k)$" is a sufficient condition for efficiently generating edge-disjoint paths. As pointed out in Section 1.2, it would be of much interest to see if this can be improved to "$y^* \leq a_1'/d$", for some constant $a_1'$.

**Acknowledgement.** We thank Alan Frieze for his helpful suggestions.

REFERENCES

[1] N. Alon, *A parallel algorithmic version of the Local Lemma*, Random Structures & Algorithms, 2 (1991), pp. 367–378.

[2] N. Alon, *The strong chromatic number of a graph*, Random Structures and Algorithms, 3 (1992), pp. 1-7.

[3] N. Alon and J. H. Spencer, *The Probabilistic Method*, Wiley–Interscience Series, John Wiley & Sons, Inc., New York, 1992.

[4] J. Beck, *An algorithmic approach to the Lovász Local Lemma*, Random Structures & Algorithms, 2 (1991), pp. 343–365.

[5] H. Chernoff, *A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations*, Annals of Mathematical Statistics, 23 (1952), pp. 493-509.

[6] A. Czumaj and C. Scheideler, *A new algorithmic approach to the General Lovász Local Lemma with applications to scheduling and satisfiability problems*, In Proc. ACM Symposium on Theory of Computing, pages 38–47, 2000.

[7] P. Erdős and L. Lovász, *Problems and results on 3-chromatic hypergraphs and some related questions*, In *Infinite and Finite Sets*, A. Hajnal et. al., eds., Colloq. Math. Soc. J. Bolyai 11, North Holland, Amsterdam, 1975, pp. 609–627.

[8] A. Frieze, *Edge-disjoint paths in expander graphs*, In Proc. ACM-SIAM Symposium on Discrete Algorithms, pages 717–725, 2000.

[9] Z. Füredi and J. Kahn, *On the dimensions of ordered sets of bounded degree*, Order, 3 (1986), pp. 15–20.

[10] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, American Statistical Association Journal, 58 (1963), pp. 13-30.

[11] J. Kleinberg, *Approximation algorithms for disjoint paths problems*, Ph. D. Thesis, Department of EECS, MIT, 1996.
[12] H. J. Karloff and D. B. Shmoys, *Efficient parallel algorithms for edge coloring problems*, Journal of Algorithms, 8 (1987), pp. 39-52.
[13] D. Knuth, *The Art of Computer Programming*, Vol. I, Addison Wesley, London, page 396 (Exercise 11), 1969.
[14] F. T. Leighton, S. B. Rao, and A. Srinivasan, *Multi-commodity flow and circuit switching*, In Proc. Hawaii International Conference on System Sciences, pages 459–465, 1998.
[15] M. Lomonosov, *Combinatorial approaches to multiflow problems*, Discrete Applied Math., 11 (1985), pp. 1–94.
[16] C.-J. Lu, *Deterministic hypergraph coloring and its applications*, In Proc. Workshop on Randomization and Approximation Techniques in Computer Science, pages 35–46, 1998.
[17] M. Middendorf and F. Pfeiffer, *On the complexity of the disjoint paths problem*, Combinatorica, 13 (1993), pp. 97–107.
[18] M. Molloy and B. Reed, *Further algorithmic aspects of the Local Lemma*, In Proc. ACM Symposium on Theory of Computing, pages 524–529, 1998.
[19] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.
[20] P. Raghavan, *Probabilistic construction of deterministic algorithms: Approximating packing integer programs*, Journal of Computer and System Sciences, 38 (1994), pp. 683-707.
[21] P. Raghavan and C. D. Thompson, *Randomized rounding: a technique for provably good algorithms and algorithmic proofs*, Combinatorica, 7 (1987), pp. 365–374.
[22] J. P. Schmidt, A. Siegel, and A. Srinivasan, *Chernoff-Hoeffding bounds for applications with limited independence*, SIAM J. Discrete Mathematics, 8 (1995), pp. 223–250.
[23] P. D. Seymour, *On odd cuts and planar multicommodity flows*, J. London Math. Soc., 42 (1981), pp. 178–192.
[24] J. H. Spencer, *The probabilistic lens: Sperner, Turán and Brégman revisited*, In *A Tribute to Paul Erdős*, A. Baker, B. Bollobás, A. Hajnal, editors, Cambridge University Press, pages 391–396, 1990.
[25] A. Srinivasan, *An extension of the Lovász Local Lemma, and its applications to integer programming*, In Proc. ACM-SIAM Symposium on Discrete Algorithms, pages 6–15, 1996.

**Appendix A. The number of rooted $u$-node ordered $d$-ary trees.** A rooted tree is called *ordered* if the relative order of each node's children is important; different relative orders give different trees. "$d$-ary" means that each tree node has at most $d$ children.

The following lemma is well-known; for an algebraic proof, see, for example, [13]. For $d = 2$, it gives a derivation of the Catalan number, which has a combinatorial proof using a reflection principle. With the help of David Mix Barrington, we generalize this idea to any $d$.

LEMMA A.1. *The number of rooted $u$-node ordered $d$-ary trees is* $\frac{1}{(d-1)u+1}\binom{du}{u}$.

*Proof.* Let $A$ denote the set of such trees. Let $B$ be the class of strings from $\{1, 2, \ldots, d\}^{du}$ generated by the following grammar:

$$T \to \lambda \mid 1\,T\,2T\,3T \cdots dT,$$

where $\lambda$ denotes the empty string. For a tree in $A$, let us add imaginary edges to nodes, including leaves, so that each node has exactly $d$ outgoing edges, labeled from 1 through $d$. Note that there are $du$ edges now. Then one can show that $|B| = |A|$ as each string in $B$ corresponds to the sequence of edge labels seen from a depth-first traversal of some tree in $A$. There is also a one-to-one correspondence between $B$ and the class $C$ of strings from $\{1, *\}^{du}$ generated by the following grammar:

$$S \to \lambda \mid 1S * S * S \cdots * S,$$

with $d - 1$ $*$'s above. Another way to see directly why $|A| = |B| = |C|$ is that they all have the same recurrence relation and the same initial condition.

Let $D$ denote the set of strings from $\{1, *\}^{du}$ with exactly $u$ 1's. As $C \subseteq D$ and $|D| = \binom{du}{u}$, it remains to calculate $|D \setminus C|$. Let $D'$ denote the set of strings from

$\{1,*\}^{du}$ with exactly $u-1$ 1's. Clearly $|D'| = \binom{du}{u-1} = \frac{u}{(d-1)u+1}\binom{du}{u}$. Next, we show that $|D \setminus C| = (d-1)|D'|$.

It is not hard to verify that $C$ contains exactly those strings in $D$ satisfying the following *prefix condition*:

- In any prefix, the number of $*$'s is at most $d-1$ times the number of 1's.

Any string in $D \setminus C$ or $D'$ must contain a prefix that first violates this condition. Let $w*$ be any such a prefix consisting of $k$ 1's and $(d-1)k+1$ $*$'s, for some $k < u$, with $w$ satisfying the prefix condition. There are exactly $\binom{du-(dk+1)}{u-k} = (d-1)\binom{du-(dk+1)}{u-k-1}$ strings in $D \setminus C$ having $w*$ as prefix, and there are exactly $\binom{du-(dk+1)}{u-k-1}$ strings in $D'$ having $w*$ as prefix. Ranging through all such strings $w*$, every string in $D \setminus C$ is enumerated exactly once, and so is every string in $D'$. So, $|D \setminus C| = (d-1)|D'|$, and $|C| = |D| - |D \setminus C| = \frac{1}{(d-1)u+1}\binom{du}{u}$. □

The following corollary follows immediately.

Corollary A.2. *In a graph of $m$ nodes and degree $d$, there are at most $s = \frac{m}{(d-1)u+1}\binom{du}{u}$ connected components of size $u$, and they can be enumerated in deterministic poly$(s)$ time.*