10-1-1997

Approximating Disjoint-Path Problems Using Greedy Algorithms and Packing Integer Programs

Stavros G. Kolliopoulos
Dartmouth College

Clifford Stein
Dartmouth College

Approximating Disjoint-Path Problems Using Greedy Algorithms and Packing Integer Programs

Stavrox Kolliopoulos
Clifford Stein

Technical Report PCS-TR97-325

11/97

Approximating Disjoint-Path Problems Using Greedy Algorithms and Packing Integer Programs

Stavros G. Kolliopoulos[1] Clifford Stein[1]

Abstract

In the *edge(vertex)-disjoint path* problem we are given a graph G and a set $\mathcal{T}$ of connection requests. Every connection request in $\mathcal{T}$ is a vertex pair (s_i, t_i), $1 \leq i \leq K$. The objective is to connect a maximum number of the pairs via edge(vertex)-disjoint paths. The edge-disjoint path problem can be generalized to the *multiple-source unsplittable flow* problem where connection request i has a demand ρ_i and every edge e a capacity u_e. All these problems are NP-hard and have a multitude of applications in areas such as routing, scheduling and bin packing.

Given the hardness of the problem, we study polynomial-time approximation algorithms. In this context, a ρ-approximation algorithm is able to route at least a $1/\rho$ fraction of the connection requests. Although the edge- and vertex-disjoint path problems, and more recently the unsplittable flow generalization, have been extensively studied, they remain notoriously hard to approximate with a bounded performance guarantee. For example, even for the simple edge-disjoint path problem, no $o(\sqrt{|E|})$-approximation algorithm is known. Moreover some of the best existing approximation ratios are obtained through sophisticated and non-standard randomized rounding schemes.

In this paper we introduce techniques which yield algorithms for a wide range of disjoint-path and unsplittable flow problems. For the general unsplittable flow problem, even with weights on the commodities, our techniques lead to the first approximation algorithm and obtain an approximation ratio that matches, to within logarithmic factors, the $O(\sqrt{|E|})$ approximation ratio for the simple edge-disjoint path problem. In addition to this result and to improved bounds for several disjoint-path problems, our techniques simplify and unify the derivation of many existing approximation results.

We use two basic techniques. First, we propose simple greedy algorithms for edge- and vertex-disjoint paths and second, we propose the use of a framework based on *packing integer programs* for more general problems such as unsplittable flow. A packing integer program is of the form maximize $c^T \cdot x$, subject to $Ax \leq b$, $A, b, c \geq 0$. As part of our tools we develop improved approximation algorithms for a class of packing integer programs, a result that we believe is of independent interest.

[1]Dartmouth College, Department of Computer Science, Hanover, NH 03755-3510. E-mail: `{stavros, cliff}@cs.dartmouth.edu`. Research partly supported by NSF Award CCR-9308701 and NSF Career Award CCR-9624828.

1 Introduction

This paper examines approximation algorithms for disjoint-path problems and their generalizations. In the *edge(vertex)-disjoint path* problem, we are given a graph $G = (V, E)$ and a set $\mathcal{T}$ of *connection requests,* also called *commodities.* Every connection request in $\mathcal{T}$ is a vertex pair (s_i, t_i), $1 \leq i \leq K$. The objective is to connect a maximum number of the pairs via edge(vertex)-disjoint paths. For vertex-disjoint paths problem, the connection requests are assumed to be disjoint. We call the set of connected pairs *realizable.* A generalization of the edge-disjoint paths problem is *multiple-source unsplittable flow.* In this problem every commodity k in the set $\mathcal{T}$ has an associated demand ρ_k, and every edge e has a capacity u_e. The demand ρ_k must be routed on a single path from s_k to t_k. The objective is to maximize the sum of the demands that can be fully routed while respecting the capacity constraints. Without loss of generality, we assume that $\max_k \rho_k = 1$, and following the standard definition of the problem in the literature, $u_e \geq 1, \forall e \in E$. When all demands and capacities are 1 in the multiple-source unsplittable flow problem we obtain the edge-disjoint path problem. (See [10, 14] for further applications and motivation for unsplittable flow.) In all the above problems one can assign a weight $w_i \leq 1$ to each connection request and seek to find a realizable set of maximum total weight. In this paper we will state explicitly when we deal with the weighted version of a problem.

Both the edge- and vertex-disjoint path problems are fundamental, extensively studied (see e.g. [25, 6, 26, 20, 10, 13, 3]), NP-hard problems [9], with a multitude of applications in areas such as telecommunications, VLSI and scheduling. Despite the attention they have received, disjoint-path problems on general graphs remain notoriously hard in terms of approximation; even for edge-disjoint paths, no algorithm is known which can find even an $\omega(1/\sqrt{|E|})$ fraction of the realizable paths.

In approximating these problems, we use the traditional notion of a *ρ-approximation algorithm,* $\rho > 1$, which is one that outputs, in polynomial time, a realizable set of size at least $1/\rho$ times the optimum. We will also give and refer to algorithms which output a realizable set whose size is non-linear function of the optimum OPT, such as $OPT^2/|E|$.

Overview of previous work. Two main approaches have been followed for approximation.
(i) The first approach, which we call the rounding approach, consists of solving a fractional relaxation and then use rounding techniques to obtain an integral solution. The fractional relaxation is typically multicommodity flow and the rounding techniques used to date involved sophisticated and non-standard use of randomized rounding [30]. The objective value of the resulting solution is compared to the *fractional optimum* y^*, which is an upper bound on the *integral optimum,* OPT. This approach has been the more successful one and recently yielded the first approximation algorithms for the edge-disjoint path problem [30], and for *uniform unsplittable flow* which is the special case of unsplittable flow where all the capacities have the same value. Let d denote the *dilation* of the fractional solution, i.e. the maximum length of a flow path in the fractional relaxation. Bounds that rely on the dilation are particularly appealing for expander graphs where it is known that $d = O(polylog(n))$ [15, 12]. The rounding approach yields, for unweighted uniform unsplittable flow (and thus for unweighted edge-disjoint paths as well) a realizable set of size $\Omega(\max\{(y^*)^2/|E|, y^*/\sqrt{|E|}, y^*/d\})$ and an $\Omega(\max\{(y^*)^2/|E|, y^*/d\})$ bound for the weighted version [30] . This approach is known to have limitations, e.g. it is known that a gap of $\Omega(\sqrt{|V|})$ exits between the fractional and integral optima for both the edge- and vertex-disjoint path problems on a graph with $|E| = O(|V|)$ [7].
(ii) Under the second approach, which we call the routing approach, a commodity is never split, i.e. routed fractionally along more than one path during the course of the algorithm. In the analysis, the objective value of the solution is compared to an estimated upper bound on the OPT. This approach has found very limited applicability so far, one reason being the perceived hardness of deriving upper bounds on OPT without resorting to a fractional relaxation. The only example of this method we are aware of is the on-line Bounded Greedy Algorithm in [10] whose approximation guarantee depends

	routing approach	rounding approach
unweighted EDP	$\frac{OPT}{\sqrt{\lvert E\rvert}}$ [10], $\frac{\mathbf{OPT}}{\sqrt{\lvert \mathbf{E_o}\rvert}}$, $\frac{\mathbf{OPT^2}}{\lvert \mathbf{E_o}\rvert}$, $\frac{\mathbf{OPT}}{\mathbf{d_o}}$	$\frac{y^*}{\sqrt{\lvert E\rvert}}$ [30], $\frac{(y^*)^2}{\lvert E\rvert}$ [30], $\frac{y^*}{d}$ [30]
weighted EDP	—	$\frac{\mathbf{y^*}}{\sqrt{\lvert \mathbf{E}\rvert}}$, $\frac{(y^*)^2}{\lvert E\rvert}$ [30], $\frac{y^*}{d}$ [30]
weighted UCUFP	—	$\frac{(y^*)^2}{\lvert E\rvert}$ [30], $\frac{y^*}{d}$ [30]
weighted UFP	—	$\frac{\mathbf{y^*}}{\log \lvert \mathbf{E}\rvert \sqrt{\lvert \mathbf{E}\rvert}}$, $\frac{(\mathbf{y^*})^2}{\lvert \mathbf{E}\rvert \log^3 \lvert \mathbf{E}\rvert}$, $\frac{\mathbf{y^*}}{\mathbf{d}}$

Table 1: Known approximation bounds for edge-disjoint paths (EDP), uniform capacity unsplittable flow (UCUFP), and general unsplittable flow (UFP), Ω-notation omitted. E_o denotes the set of edges used by some path in the integral optimal solution and d_o the average length of the paths in the same solution. Results with no citation come from the present paper. Our $y^*/\sqrt{|E|}$ bound for the weighted EDP problem holds under the assumption that the number of connection requests $K = O(|E|)$.

also on the diameter of the graph. The algorithm can be easily modified into an off-line procedure that outputs realizable sets of size $\Omega(OPT/\sqrt{|E|})$ ($\Omega(OPT/\sqrt{|V|})$) for edge(vertex)-disjoint paths. The $\Omega(OPT/\sqrt{|V|})$ bound is the best known bound to date for vertex-disjoint paths.

Our contribution. In this paper we provide techniques for approximating disjoint-path problems that bear on both of the above approaches. Tables 1 and 2 summarize previous and new bounds for edge-, vertex-disjoint path and unsplittable flow problems.

Under the routing approach (approach **(ii)**) we give a simple deterministic greedy algorithm Greedy_Path for edge-disjoint paths that has performance guarantees comparable to those obtained by the multicommodity flow based algorithms. Greedy algorithms have been extensively studied in combinatorial optimization due to their elegance and simplicity. Our work provides another example of the usefulness of the greedy method. The underlying idea is that if one keeps routing commodities along sufficiently short paths the final number of commodities routed is lowerbounded with respect to the optimum.

Greedy_Path outputs a realizable set of size $\Omega(\max\{OPT^2/|E_o|, OPT/\sqrt{|E_o|}\})$ for the edge-disjoint path problem. Here $E_o \subseteq E$ is the set of edges used by the paths in an optimal solution. Note that $OPT^2/|E_o|$ always dominates $OPT/\sqrt{|E_o|}$ in the unweighted case that we consider; we give both bounds to facilitate comparison with existing work and to conform with the traditional notion of a ρ-approximation algorithm. Our approximation existentially improves upon the multicommodity-flow based results when $|E_o| = o(|E|)$, i.e. when the optimal solution uses a small portion of the edges of the graph. Another bound can be obtained by noticing that $OPT^2/|E_o| = OPT/d_o$, where d_o denotes the average length of the paths in an optimal solution.

Essentially the same algorithm, Greedy_VPath, obtains for the vertex-disjoint path problem a realizable set of size $\Omega(\max\{OPT^2/|V_o|, OPT/\sqrt{|V_o|}\})$, where $V_o \subseteq V$ is the set of vertices used by the paths in an optimal solution. Recall that the best known bound to date is $t = \Omega(OPT/\sqrt{|V|})$. The realizable set output by our algorithm has size $\Omega(t^2)$ and potentially better than this when $|V_o| = o(|V|)$. This is a significant improvement when $OPT = \omega(\sqrt{|V|})$. For example, when $OPT = \Omega(|V|)$, we obtain a constant-factor approximation. Again an $\Omega(OPT/d_o)$ guarantee follows immediately.

We turn to the rounding approach (approach **(i)**) to handle the weighted disjoint path and unsplittable flow problems. We propose the use of *packing integer programs* as a unifying framework that abstracts away the need for customized and complex randomized rounding schemes. A *packing integer program* is of the form maximize $c^T \cdot x$, subject to $Ax \leq b$, $A, b, c \geq 0$. We first develop, as part of our tools, an improved approximation algorithm for a class of packing integer programs, called column restricted, that are relevant to unsplittable flow problems. Armed with both this new algorithm and

	routing approach	rounding approach
unweighted	$\frac{OPT}{\sqrt{\lvert V\rvert}}$ [10], $\frac{\mathbf{OPT}}{\sqrt{\lvert \mathbf{V_o}\rvert}}$, $\frac{\mathbf{OPT}^2}{\lvert \mathbf{V_o}\rvert}$, $\frac{\mathbf{OPT}}{\mathbf{d_o}}$	$\frac{\mathbf{y}^*}{\sqrt{\lvert \mathbf{V}\rvert}}$, $\frac{(\mathbf{y}^*)^2}{\lvert \mathbf{V}\rvert}$, $\frac{\mathbf{y}^*}{\mathbf{d}}$
weighted	—	$\frac{\mathbf{y}^*}{\sqrt{\lvert \mathbf{V}\rvert}}$, $\frac{(\mathbf{y}^*)^2}{\lvert \mathbf{V}\rvert}$, $\frac{\mathbf{y}^*}{\mathbf{d}}$

Table 2: Known approximation bounds for vertex-disjoint paths, Ω-notation omitted. V_o denotes the set of vertices used by some path in the integral optimal solution and d_o the average length of the paths in the same solution. Results with no citation come from the present paper.

existing algorithms for general packing integer programs, we show how packing formulations both provide a unified and simplified derivation of many results from [30] and lead to new ones. In particular, we obtain the first approximation algorithm for weighted multiple-source unsplittable flow on networks with arbitrary demands and capacities and the first approximation algorithm for weighted vertex-disjoint paths. Further, we believe that our new algorithm for column-restricted packing integer programs is of independent interest.

We now elaborate on our results under the rounding approach, providing further background as necessary.

1.1 Packing Integer Programs

Packing integer programs are a well-studied class of integer programs that can model several NP-complete problems, including independent set, hypergraph k-matching [18, 1], job-shop scheduling [22, 27, 32, 19] and many flow and path related problems. Many of these problems seem to be difficult to approximate, and not much is known about their worst-case approximation ratios. Following [29] a packing integer program (PIP) is defined as follows.

Definition 1.1 *Given $A \in [0,1]^{m\times n}$, $b \in [1,\infty)^m$ and $c \in [0,1]^n$ with $\max_j c_j = 1$, a PIP $\mathcal{P} = (A,b,c)$ seeks to maximize $c^T \cdot x$ subject to $x \in Z_+^n$ and $Ax \leq b$. Constraints of the form $0 \leq x_j \leq d_j$ are also allowed. If $A \in \{0,1\}^{m\times n}$, each entry of b is assumed integral. Let $B = \min_i b_i$, and α be the maximum number of non-zero entries in any column of A.*

The parameters B and α in the definition above appear in the approximation bounds. For convenience we call b_i the *capacity* of row i. The restrictions on the values of the entries of A, b, c are without loss of generality; the values in an arbitrary packing program can be scaled to satisfy the above requirements [28]. We will state explicitly when some packing program in this paper deviates from these requirements. When $A \in \{0,1\}^{m\times n}$, we say that we have a $(0,1)$-PIP.

Previous Work on Packing Programs. The basic techniques for approximating packing integer programs have been the *randomized rounding* technique of Raghavan and Thompson [23, 24] and the work of Plotkin, Shmoys and Tardos [22]. Let y^* denote the optimum value of the linear relaxation. Standard randomized rounding yields integral solutions of value $\Omega(y^*/m^{1/B})$ for general PIP's and $\Omega(y^*/m^{1/(B+1)})$ for $(0,1)$-PIP's [24] (see also [28].) Srinivasan [28, 29] improved on the standard randomized rounding bounds and obtained bounds of $\Omega(y^*(y^*/m)^{1/(B-1)})$ and $\Omega(y^*/\alpha^{1/(B-1)})$ for general PIP's and $\Omega(y^*(y^*/m)^{1/B})$ and $\Omega(y^*/\alpha^{1/B})$ for $(0,1)$-PIP's.

New results for column-restricted PIP's. The above results show that for various combinations of values for y^*, m and B, the bounds obtained for a $(0,1)$-PIP are significantly better than those for general PIP's. In fact they are always better when $y^* < m$. As another example, the approximation ratio $m^{1/(B+1)}$ obtained for a $(0,1)$-PIP is *polynomially better* than the approximation ratio of a PIP with the same parameters. Thus it is natural to ask whether we can bridge this gap. We make progress in this direction by defining a *column-restricted* PIP $\mathcal{P}_r$ as one where all non-zero entries of the j-th column of A have the same value $\rho_j \leq 1$. Column-restricted PIP's arise in applications such as unsplittable flow problems (see next section). We show how to obtain approximation guarantees for column-restricted PIP's that are similar to the ones obtained for $(0,1)$-PIP's. Let y_r^* denote the optimum of the linear relaxation of $\mathcal{P}_r$. We obtain an integral solution of value $\Omega(y_r^*/m^{1/B})$ and $\Omega(y_r^*/\alpha^{1/B})$. Letting $\sigma(y_r^*) = \Omega(y_r^*(y_r^*/m)^{1/B})$ we also obtain a bound that is at least as good as $\sigma(y_r^*)$ for $y_r^* < m \log n$ and in any case it is never worse by more than a $O(\log^{1/B} n)$ factor. Finally we show how to improve upon the stated approximations when $\max_j \rho_j$ is bounded away from 1.

We now give an overview of our technique. First we find an optimum solution x^* to the linear relaxation of the column-restricted PIP $\mathcal{P}_r$. We partition the ρ_j's into a fixed number of intervals according to their values and generate a packing subproblem for each range. In a packing subproblem $\mathcal{P}^L$ corresponding to range L, we only include the columns of A with $\rho_j \in L$ and to each component of the b^L-vector we allocate only a fraction of the original b_i value, a fraction that is determined by x^*. Next we find approximate solutions to each subproblem and combine them to obtain a solution to the original problem. Perhaps the key idea is in using the solution x^* to define the capacity allocation to the b^L-vector for subproblem $\mathcal{P}^L$. This generalizes previous work of the authors [14] on single-source unsplittable flow. The other key idea is that each subproblem can be approximated almost as well as a $(0,1)$-PIP.

1.2 Applications of packing to approximation

We introduce a new framework for applying packing techniques to disjoint-path problems. First, we formulate an integer program (which is not necessarily a PIP) and solve a linear relaxation of this integer program to obtain a solution x. Typically this is a multicommodity flow problem. We then explicitly use the solution x to guide the formation of a column-restricted or $(0,1)$ PIP. A related usage of a solution to the linear relaxation of integer programs in a different context can be found in [8, 31]. An integral approximate solution to the created PIP will be an approximate solution to the original disjoint path problem (with possibly some small degradation in the approximation factor). This integral solution can be found using existing algorithms for approximating PIP's as a black box. Our algorithms apply to the case when there are weights on the commodities, and thus generalize those of Srinivasan for edge-disjoint paths. This approach yields four applications which we explain below.

Application 1: weighted unsplittable flow. Let F_1, F_2, F_3 denote $(y^*)^2/|E|, y^*/\sqrt{|E|}$ and y^*/d respectively. We obtain a realizable set of weight $\Omega(\max\{F_2/\log|E|, F_3\})$ for unsplittable flow with arbitrary demands and capacities. We also give a bound at least as good as $F_1/\log^3|E|$ whose analytical form is complicated (cf. Thms 3.1,4.4.) In the case where the number of commodities $K = O(|E|)$ we show how to obtain an $\Omega(\max\{F_1/\log|E|, F_2\})$ bound. Notice that for the edge-disjoint path problem this is a natural assumption since at most $|E|$ connection requests can be feasibly routed. We also note that a ρ-approximation for y^* entails an $O(\rho \log |E|)$ approximation for the problem of *routing in rounds* [2, 10]. We do not pursue any further the latter problem in this extended abstract.

Application 2: weighted vertex-disjoint paths. We give an algorithm that outputs a solution of value $\Omega(\max\{(y^*)^2/|V|, y^*/\sqrt{|V|}, y^*/d\})$. The algorithm relies on the observation that after solving the relaxation the problem of rounding is essentially an instance of hypergraph matching; thus it can be formulated as a packing program with $|V|$ constraints. The algorithm is surprisingly simple but the performance guarantee matches the integrality gap known for the problem [7].

Application 3: routing with low congestion. A problem that has received a lot of attention in the literature on routing problems (e.g. [24, 16, 22, 21, 10, 14]) is that of minimizing *congestion*, i.e. the factor by which one is allowed to scale up capacities in order to achieve an optimal (or near-optimal) realizable set. In our usage of packing in the rounding algorithms we have assumed that the parameter B of the packing program is equal to 1. Allowing $B > 1$ is equivalent to allowing congestion B in the corresponding disjoint-path problem. Thus another advantage of the packing approach is that tradeoffs with the allowed congestion B can be obtained immediately by plugging in B in the packing algorithms that we use as a black box. For example the approximation for edge-disjoint paths becomes $\Omega(\max\{y^*(y^*/|E|)^{1/B}, y^*/|E|^{1/(B+1)}, y^*/d^{1/B}\})$, when the number of connection requests is $O(|E|)$. Our congestion tradeoffs generalize previous work by Srinivasan [30] who showed the $\Omega(y^*/d^{1/B})$ tradeoff for uniform capacity unsplittable flow. We do not state the tradeoffs explicitly for the various problems since they can be obtained easily by simple modifications to the given algorithms.

Application 4: Independent Set in the square of a graph. Given a graph $G = (V, E)$ the *k-th power* $G^k = (V, E^k)$ of G is a graph where two vertices are adjacent if and only if they are at distance at most k in G. We further demonstrate the power of packing formulations by providing an $O(\sqrt{|V|})$ approximation algorithm for finding a maximum independent set in the square of a graph. We also give results that depend on the maximum vertex degree Δ in G. Our approximation ratio cannot be polynomially improved in the sense that no $(n/4)^{1/2-\varepsilon}$ approximation, for any fixed $\varepsilon > 0$, can be obtained in polynomial time unless $NP = ZPP$. Studying NP-hard problems in powers of graphs is a topic that has received some attention in the literature [5, 33, 17, 4].

Independently of our work Srinivasan (personal communication) has obtained results similar to ours for approximating vertex-disjoint paths under the rounding approach and column-restricted packing integer programs. His work builds on the methods in [30].

2 Approximating a column-restricted PIP

In this section we present the approximation algorithm for column-restricted PIP's. Let $\mathcal{P} = (A, b, c)$ be a column-restricted PIP. We call $\rho_j \leq 1$, the value of the non-zero entries of the j-th column, $1 \leq j \leq n$, the *value* of column j. Throughout this section we assume that there is a polynomial-time algorithm that given a $(0,1)$-PIP with fractional optimum y^* outputs an integral solution of value at least $\sigma(m, B, \alpha, y^*)$ where m, B, α are the parameters of the packing program. For example a known σ is $\Omega(y^*/m^{1+B})$. We start by providing a subroutine for solving a column-restricted PIP when the column values are close in range.

Theorem 2.1 *Let $\mathcal{P} = (A, b, c)$ be a column-restricted PIP where all column values ρ_j are equal to ρ and where each $b_i = k_i\Gamma\rho$, $\Gamma \geq 1$, k_i positive integer, $1 \leq i \leq m$. Here $\min_i b_i$ is not necessarily greater than 1. Then we can find in polynomial time a solution of value at least $\sigma(m, k\Gamma, \alpha, y^*)$, where y^* denotes the optimum of the linear relaxation of $\mathcal{P}$ and $k = min_i k_i$.*

Proof. Transform the given system $\mathcal{P}$ to a $(0,1)$-PIP $\mathcal{P}' = (A', b', c)$ where $b'_i = k_i\Gamma$, and $A'_{ij} = A_{ij}/\rho$. Every feasible solution (either fractional or integral) $\bar{x}$ to $\mathcal{P}'$ is a feasible solution to $\mathcal{P}$ and vice versa. Therefore the fractional optimum y^* is the same for both programs. Also the maximum number of non-zero entries on any column is the same for A and A'. Thus we can unambiguously use α for both. We have assumed that there is an approximation algorithm for $\mathcal{P}'$ returning a solution with objective value $\sigma(m, k\Gamma, \alpha, y^*)$. Invoking this algorithm completes the proof. ▌

The proof of the following lemma generalizes that of Lemma 4.1 in [11].

Lemma 2.1 *Let $\mathcal{P} = (A, b, c)$, be a column-restricted PIP with column values in the interval $(a_1, a_2]$, and $b_i \geq \Gamma a_2$, $\forall i$ and some number $\Gamma \geq 1$. Here $\min_i b_i$ is not necessarily greater than 1. There is an algorithm α_PACKING that finds a solution g to $\mathcal{P}$ of value at least $\sigma(m, \Gamma, \alpha, \frac{a_1}{2a_2} y^*)$, where y^* is the optimum of the fractional relaxation of $\mathcal{P}$. The algorithm runs in polynomial time.*

Proof sketch. We sketch the algorithm α_PACKING. Obtain a PIP $\mathcal{P}' = (A', b', c)$ from $\mathcal{P}$ as follows. Round down b_i to the nearest multiple of Γa_2 and then multiply it with a_1/a_2. Set b'_i equal to the resulting value. Every b'_i is now between $a_1/2a_2$ and a_1/a_2 times the corresponding b_i. Set A'_{ij} to a_1 if $A_{ij} \neq 0$ and to 0 otherwise. $\mathcal{P}'$ has thus a fractional solution of value at least $(a_1/2a_2)y^*$ that can be obtained by scaling down the optimal fractional solution of $\mathcal{P}$. Note that every b'_i is a multiple of Γa_1. Thus we can invoke Theorem 2.1 and find a solution g' to $\mathcal{P}'$ of value at least $\sigma(m, \Gamma, \alpha, (a_1/2a_2)y^*)$. Scaling up every component of g' by a factor of at most a_2/a_1 yields a vector g that is feasible for $\mathcal{P}$ and has value at least $\sigma(m, \Gamma, \alpha, \frac{a_1}{2a_2} y^*)$. ▌

Lemma 2.2 *Let $\mathcal{P} = (A, b, c)$, be a column-restricted PIP with column values in the interval $(a_1, a_2]$, and b_i a multiple of Γa_2, $\forall i$ and some number $\Gamma \geq 1$. Here $\min_i b_i$ is not necessarily greater than 1. There is an algorithm INTERVAL_PACKING that finds a solution g to $\mathcal{P}$ of value at least $\sigma(m, \Gamma, \alpha, \frac{a_1}{a_2} y^*)$, where y^* is the optimum of the fractional relaxation of $\mathcal{P}$. The algorithm runs in polynomial time.*

Proof sketch. Similar to that of Lemma 2.1. Since all b_i's are already multiples of Ba_2, we don't pay the 1/2 factor for the initial rounding. ▌

We now give the idea behind the full algorithm. The technique generalizes earlier work of the authors on single-source unsplittable flow [14]. Let x^* denote the optimum solution to the linear relaxation of $\mathcal{P}$. We are going to create packing subproblems $P^\lambda = (A^\lambda, b^\lambda, c^\lambda)$ where A^λ contains only the columns of A with values in some fixed range $(\alpha_{\lambda-1}, \alpha_\lambda]$. We will obtain our integral solution to $\mathcal{P}$ by combining approximate solutions to the subproblems. The crucial step is capacity allocation to subproblems. Consider a candidate for the b^λ-vector that we call the λ-th *fractional capacity vector.* In the fractional capacity vector the i-th entry is equal to $\sum_{j|\rho_j \in (\alpha_{\lambda-1}, \alpha_\lambda]} A_{ij} x^*_j$. In other words we allocate capacity to the λ-th subproblem by pulling out of b the amount of capacity used up in the solution x^* by the columns with values in $(\alpha_{\lambda-1}, \alpha_\lambda]$. The benefit of such a scheme would be that by using the relevant entries of x^* we could obtain a feasible solution to the linear relaxation of P^λ. However, to be able to benefit from Lemma 2.1 to find an approximate integral solution to each P^λ, we need all entries of b^λ to be larger than $B\alpha_\lambda$. This increases the required capacity for each subproblem potentially above the fractional capacity. Thus we resort to more sophisticated capacity allocation to ensure that (i) there is enough total capacity in the b-vector of $\mathcal{P}$ to share among the subproblems (ii) the subproblems can benefit from the subroutines in Lemmata 2.1, 2.2. In particular, we initially assign to each subproblem λ only 1/2 of the fractional capacity vector; this has the asymptotically negligible effect of scaling down the fractional optimum for each subproblem by at most 2. We exploit the unused $(1/2)b$ vector of capacity to add an extra $B\alpha_\lambda$ units to the entries of b^λ, $\forall \lambda$.

Given α_i, α_j, let J^{α_i, α_j} be the set of column indices k for which $\alpha_i < \rho_k \leq \alpha_j$. We then define, A^{α_i, α_j} to be the $m \times |J^{\alpha_i, \alpha_j}|$ submatrix of A consisting of the columns in J^{α_i, α_j}, and for any vector x, x^{α_i, α_j} to be the $|J^{\alpha_i, \alpha_j}|$-entry subvector x consisting of the entries whose indices are in J^{α_i, α_j}. We will also need to combine back together the various subvectors, and define $x^{\alpha_1, \alpha_2} \cup \cdots \cup x^{\alpha_{k-1}, \alpha_k}$ to be the n-entry vector in which the various entries in the various subvectors are mapped back into their original positions. Any positions not in any of the corresponding index sets J^{α_i, α_j} are set to 0.

ALGORITHM COLUMN_PARTITION($\mathcal{P}$)

Step 1. Find the n-vector x_* that yields the optimal solution to the linear relaxation of $\mathcal{P}$.

Step 2a. Define a partition of the $(0, 1]$ interval into $\xi = O(\log n)$ consecutive subintervals $(0, n^{-k}], \ldots,$ $(4^{-\lambda}, 4^{-\lambda+1}], \ldots, (4^{-2}, 4^{-1}], (4^{-1}, 1]$ where k is a constant larger than 1. For $\lambda = 1 \ldots \xi - 1$ form subproblem $P^\lambda = (A^\lambda, b^\lambda, c^\lambda)$. A^λ and c^λ are the restrictions defined by $A^\lambda = A^{4^{-\lambda}, 4^{-\lambda+1}}$ and $c^\lambda = c^{4^{-\lambda}, 4^{-\lambda+1}}$. Define similarly $P^\xi = (A^\xi, b^\xi, c^\xi)$ such that $A^\xi = A^{0, n^{-k}}$, $c^\xi = c^{0, n^{-k}}$.

Step 2b. Let d_i^λ be the ith entry of $(A^\lambda \cdot x^\lambda)$. We define $b^\lambda = b$ when λ is 1 or ξ, and otherwise $b_i^\lambda = B(1/4)^{\lambda-1} + (1/2)d_i^\lambda$.

Step 3. Form solution $\dot{x}$ by setting x_j to 1 if $\rho_j \in (0, n^{-k}]$, and 0 otherwise.

Step 4. On each P^λ, $2 \leq \lambda \leq \xi - 1$, invoke α_PACKING to obtain a solution vector x^λ. Combine the solutions to subproblems 2 through $\xi - 1$ to form n-vector $\hat{x} = \cup_{2 \leq \lambda \leq \xi-1} x^\lambda$.

Step 5. Invoke INTERVAL_PACKING on P^1 to obtain a solution vector x^1. Let $\bar{x} = \cup x^1$.

Step 6. Of the three vectors $\dot{x}$, $\hat{x}$ and $\bar{x}$, output the one, call it x, that maximizes $c^T \cdot x$.

Two tasks remain. First, we must show that the vector x output by the algorithm is a feasible solution to the original packing problem $\mathcal{P}$. Second, we must lower bound $c^T \cdot x$ in terms of the optimum $y_* = c^T \cdot x_*$ of the fractional relaxation of $\mathcal{P}$. Let $y_*^{r_1,r_2} = c^{r_1,r_2} x_*^{r_1,r_2}$, and if $r_1 = (1/4)^\lambda$ and $r_2 = (1/4)^{\lambda-1}$ then we abbreviate $y_*^{r_1,r_2}$ as y_*^λ. We examine first the vector $\hat{x}$ in the following lemma.

Lemma 2.3 *Algorithm* COLUMN_PARTITION *runs in polynomial time and the n-vector $\hat{x}$ it outputs is a feasible solution to $\mathcal{P}$ of value at least* $\sum_{\lambda=2}^{\lambda=\xi-1} \sigma(m, B, \alpha, (1/16)y_*^\lambda)$.

Proof sketch. Let $\hat{y}^\lambda$ be the optimal solution to the linear relaxation of P^λ. By Lemma 2.1 the value of the solution x^λ, $2 \leq \lambda \leq \xi - 1$, found at Step 4 is at least $\sigma(m, B, \alpha, (1/2)(1/4)\hat{y}^\lambda)$. By the definition of b^λ in P_λ, $(1/2)x_*^\lambda$,i.e. the restriction of x scaled by $1/2$, is a feasible fractional solution for P_λ. Thus $\hat{y}^\lambda \geq (1/2)y_*^\lambda$. Hence the value of x^λ is at least $\sigma(m, B, \alpha, (1/16)y_*^\lambda)$. The claim on the value follows.

For the feasibility, we note that the aggregate capacity used by $\hat{x}$ on row i of A is the sum of the capacities used by x^λ, $2 \leq \lambda \leq \xi - 1$, on each subproblem. This sum is by Step 2b at most $(1/2)\sum_{\lambda=2}^{\lambda=\xi-1} d_i^\lambda + B\sum_{\lambda=2}^{\lambda=\xi-1}(1/4)^{\lambda-1}$. But $B\sum_{\lambda=2}^{\xi-1}(1/4)^{\lambda-1} < 1/2 < (1/2)b_i$ and $\sum_{\lambda=2}^{\lambda=\xi-1} d_i^\lambda \leq b_i$. Thus the aggregate capacity used by $\hat{x}$ is at most $(1/2)b_i + (1/2)b_i = b_i$. ∎

It remains to account for $\bar{x}$ and $\dot{x}$ The following theorem is the main result of this section.

Theorem 2.2 *Let $\mathcal{P} = (A, b, c)$ be a column-restricted PIP and y^* be the optimum of the linear relaxation of $\mathcal{P}$. Algorithm* COLUMN_PARTITION *finds in polynomial time a solution g to $\mathcal{P}$ of value* $\Omega(\max\{y^*/m^{1/(B+1)}, y^*/\alpha^{1/B}, y^*(y^*/m\log n)^{1/B}\})$.

Proof. Each of the three vectors $\dot{x}, \hat{x}, \bar{x}$ solves a packing problem with demands lying in $(0, 1/n^k]$, $(1/n^k, 1/4]$ and $(1/4, 1]$ respectively. Let $\dot{\mathcal{P}}, \hat{\mathcal{P}}, \bar{\mathcal{P}}$ be the three induced packing problems. The optimal solution to the linear relaxation of at least one of them will have value at least 1/3 of the optimal solution to the linear relaxation of $\mathcal{P}$. It remains to lower bound the approximation achieved by each of the three vectors on its corresponding domain of demands. Since m, B, and α are fixed for all subproblems, note that σ is a function of one variable, y^*.

Vector $\dot{x}$ solves $\dot{\mathcal{P}}$ optimally. The solution is feasible since all demands are less than $1/n^k$ and thus the value of the left-hand side of any packing constraint cannot exceed 1. By Lemma 2.1 vector $\bar{x}$ outputs a solution to $\bar{\mathcal{P}}$ of value at least $\sigma(m, B, \alpha, (1/4)y_*^{1/4,1})$.

For $\hat{\mathcal{P}}$, the value of the solution output is given by the sum $A = \sum_{\lambda=2}^{\xi-1} \sigma(m, B, \alpha, (1/16)y_*^\lambda)$ in Lemma 2.3. We distinguish two cases. If σ is a function linear in y_* then $A \geq \sigma(m, B, \alpha, (1/16)y_*^{n^{-k},4^{-1}})$. If σ is a function convex in y_* the sum A is minimized when all the terms y_*^λ are equal to $\Theta(y_*^{n^{-k},4^{-1}} / \log n)$. Instantiating $\sigma(y_*)$ with the function $\Omega(\max\{y_*/m^{1/(B+1)}, y_*/\alpha^{1/B}\})$ in the linear case and with $\Omega(y_*(y_*/m)^{1/B})$ in the convex case completes the proof. ∎

We can actually obtain an $\omega(y^*(y^*/m\log n)^{1/B})$ bound for $y^* < m\log n$ that has a complicated analytic expression. The details are in the Appendix. Based on this improved bound and a different parameterization of the algorithm, we can obtain the following result for the case when the maximum column value $\max_j \rho_j = \rho$ is bounded away from 1.

Theorem 2.3 *Let $\mathcal{P} = (A, b, c)$ be a column-restricted PIP with $\rho < 1$ being the maximum column value. Let δ_2, $0 < \delta_2 < 1$, be a constant such that $(1-\rho)(1-\delta_2)/\rho \geq 1$. Let y^* be the optimum of the linear relaxation of $\mathcal{P}$ and ζ be any constant greater than or equal to $(1-\rho)(1-\delta_2)/\rho$. There is a polynomial time algorithm that outputs a solution to $\mathcal{P}$ of value $\Omega(y^*(y^*/m \log n)^{1/(\zeta B)})$.*

3 Applications of PIP's to approximation

3.1 Weighted multiple-source unsplittable flow

In this section we examine the weighted multiple-source unsplittable flow problem.

Our approach consists of finding the optimum of the fractional relaxation, i.e. weighted multicommodity flow, which can be solved in polynomial time via linear programming. The relaxation consists of allowing commodity k to be shipped along more than one path. Call these paths the *fractional paths.* We round in two stages. In the first stage we select at most one of the fractional paths for each commodity, at the expense of congestion, i.e. some capacities may be violated. In addition, some commodities may not be routed at all. In the second stage, among the commodities routed during the first stage, we select those that will ultimately be routed while respecting the capacity constraints. It is in this last stage that a column-restricted PIP is used.

We introduce some terminology before giving the algorithm. A *routing* is a set of s_{k_i}-t_{k_i} paths P_{k_i}, used to route ρ_{k_i} amount of flow from s_{k_i} to t_{k_i} for each $(s_{k_i}, t_{k_i}) \in I \subseteq \mathcal{T}$. Given a routing g, the flow g_e through edge e is equal to $\sum_{P_i \in g, P_i \ni e} \rho_i$. A routing g for which $g_e \leq u_e$ for every edge e is an *unsplittable flow.* A *fractional routing* is one where the flow for some commodity is split on potentially many paths; this is just standard multicommodity flow. A *fractional single-path routing* is one where the flow for a commodity is shipped on one path if at all, but only a fraction of the demand ρ_k is shipped for commodity k. The value of a routing g is the weighted sum of the demands routed in g.

ALGORITHM MAX_ROUTING$(G = (V, E, u), \mathcal{T})$

Step 1. Find an optimum fractional routing f by invoking a weighted multicommodity flow algorithm. Denote by $\alpha^f(\mathcal{T})$ the value of f.

Step 2. Scale up all capacities by a factor of $\Theta(\log |E|)$ to obtain network G' with capacity function u'. Invoke Raghavan's algorithm [23] on G' to round f to a routing g'.

Step 3. Scale down the flow on every path of g' by a factor of at most $\Theta(\log |E|)$ to obtain a fractional single-path routing g that is feasible for G.

Step 4. Construct a column-restricted PIP $\mathcal{P} = (A, b, c)$ as follows. Let $k_1, k_2, \ldots, k_\lambda$ be the set of commodities shipped in g, $\lambda \leq K$. A has λ columns, one for each commodity in g, and $|E|$ rows, one for each edge of G. $A_{ij} = \rho_{k_j}$ if the path P_{k_j} in g for commodity k_j goes through edge e_i and 0 otherwise. The cost vector c has entry c_j set to w_{k_j} for each commodity k_j shipped in g. Finally, $b_i = u_{e_i}$, $1 \leq i \leq |E|$.

Step 5. Invoke algorithm COLUMN_PARTITION to find an integral solution $\hat{g}$ to $\mathcal{P}$. Construct an unsplittable flow g'' by routing commodity k_j on path P_{k_j} if and only if $\hat{g}_j = 1$.

Theorem 3.1 *Given a weighted multiple-source unsplittable flow problem $(G = (V, E), \mathcal{T})$, algorithm* MAX_ROUTING *finds in polynomial time an unsplittable flow of value at least $\Omega(\max\{\alpha^f(\mathcal{T})/(\log |E| \sqrt{|E|}), (\alpha^f(\mathcal{T}))^2/(|E| \log^3 |E|)\})$, where $\alpha^f(\mathcal{T})$ is the value of an optimum fractional routing f.*

Proof sketch. First note that since routing g' is feasible for G', after scaling down the flow at Step 3, routing g is feasible for G. The rounded solution to $\mathcal{P}$ maintains feasibility. For the value, we can easily extend the analysis of Raghavan's algorithm to show that even with weights, it routes in G' a constant fraction of $\alpha^f(\mathcal{T})$ [23]. Set z_{k_j} to be equal to the amount of flow that is routed in g for commodity k_j divided by ρ_{k_j}. The λ-vector z is a feasible fractional solution to $\mathcal{P}$ of value $\Omega(\alpha^f(\mathcal{T})/\log |E|)$. The theorem follows by using Theorem 2.2 to lower bound the value of the integral solution $\hat{g}$ to $\mathcal{P}$. ∎

We now give a bound that depends on the dilation d of the fractional routing f.

Theorem 3.2 *Given a weighted multiple-source unsplittable flow problem $(G = (V, E), \mathcal{T})$, there is a polynomial-time algorithm that finds an unsplittable flow of value at least $\Omega(\alpha^f(\mathcal{T})/d)$, where $\alpha^f(\mathcal{T})$ is the value and d the dilation of an optimum fractional routing f.*

Proof. See Appendix. ∎

The construction in the proof of Theorem 3.2 can also be used to give an $\Omega(\max\{\alpha^f(\mathcal{T})/(\sqrt{|E|}), (\alpha^f(\mathcal{T}))^2/(|E|\log|E|)\})$, bound in the case where the number of commodities $|\mathcal{T}| = O(|E|)$. We omit the details.

3.2 Weighted vertex-disjoint paths

In this section we give an approximation algorithm for the weighted vertex-disjoint path problem.

A relaxation of the problem is integral multicommodity flow where every commodity has an associated demand of 1. We can express vertex-disjoint paths as an integer program $\mathcal{I}$ that has the same constraints as multicommodity flow together with additional "bandwidth" constraints on the vertices such that the total flow through a vertex is at most 1. Let $\mathcal{LP}$ be the linear relaxation of $\mathcal{I}$. The optimal solution f to $\mathcal{LP}$ consists of a set of *fractional flow paths.* Our algorithm relies on the observation that f gives rise (and at the same time is a fractional solution) to a PIP. The particular PIP models a 1-matching problem on a hypergraph H with vertex set V and a hyperedge (subset of V) for every path in the fractional solution. In other words, the paths in f may be viewed as sets of vertices *without any regard for the flow through the edges of G.* We proceed to give the full algorithm.

ALGORITHM PATH_PACKING$(G = (V, E), \mathcal{T})$

Step 1. Formulate the linear relaxation $\mathcal{LP}$ and find an optimal solution f to it. Using flow decomposition express f as a set of paths $P_1, P_2, \ldots, P_\lambda$ each connecting a pair of terminals and carrying $z_i \leq 1$, $1 \leq i \leq \lambda$, units of flow.

Step 2. Construct a $(0,1)$-PIP $\mathcal{P} = (A, b, c)$ as follows. A is a $|V| \times \lambda$ matrix; A_{ij} is one if path P_j includes vertex i and 0 otherwise. b is a vector of ones. c_j is equal to w_k such that path P_j connects terminals s_k and t_k.

Step 3. Find an integral solution g to $\mathcal{P}$ and output the corresponding set of paths $P(g)$.

Theorem 3.3 *Given a weighted vertex-disjoint paths problem $(G = (V, E), \mathcal{T})$, algorithm* PATH_PACKING *finds in polynomial time a solution of value $\Omega(\max\{y^*/\sqrt{|V|}, (y^*)^2/|V|, y^*/d\})$, where d is the dilation and y^* is the value of an optimum solution to the linear relaxation $\mathcal{LP}$.*

Proof sketch. We show first that $P(g)$ is a feasible solution to the problem. Clearly the constraints of the packing program $\mathcal{P}$, constructed at Step 2 ensure that the paths in $P(g)$ are vertex disjoint. This excludes the possibility that more than one (s_k, t_k)-path is present in $P(g)$ for some k. The optimal value of the linear relaxation of $\mathcal{P}$ is at least y^* since setting x_j equal to z_j, $1 \leq j \leq \lambda$, yields a feasible fractional solution to $\mathcal{P}$. By applying either standard randomized rounding [24] or Srinivasan's algorithms [28, 29] at Step 3, we obtain the claimed bounds on the objective value $c^T \cdot g$. ∎

3.3 An Application to Independent Set

In this section we show how a packing formulation leads to an $O(\sqrt{|V|})$ approximation for the following problem: find a maximum weight independent set in the square of the graph $G = (V, E)$.

Theorem 3.4 *Given a graph $G = (V, E)$ and $c \in [0, 1]^{|V|}$ a weight vector on the vertices, there exists a polynomial-time algorithm that outputs an independent set in the square $G^2 = (V, E^2)$ of G of weight $\Omega(\max\{y^*/\sqrt{|V|}, (y^*)^2/|V|, y^*/\Delta\})$. Here y^* denotes the optimum of a fractional relaxation and Δ is the maximum vertex degree in G.*

A hardness of approximation result for the problem of finding a maximum independent set in the k-th power of a graph follows.

Theorem 3.5 *For the problem of finding a maximum independent set in the k-th power $G^k = (V, E^k)$ of a graph $G = (V, E)$ for any fixed integer $k > 0$, there is no ρ-approximation with $\rho = (\frac{|V|}{k+2})^{1/2-\varepsilon}$, for any fixed $\varepsilon > 0$, unless $NP = ZPP$.*

4 Greedy Algorithms for Disjoint Paths

In this section we turn to the routing approach for the unweighted edge- and vertex-disjoint path problems.

```
ALGORITHM GREEDY_PATH($G, \mathcal{T}$)
Step 1. Set $\mathcal{A}$ to $\emptyset$.
Step 2. Let $(s_*, t_*)$ be the commodity in $\mathcal{T}$ such that the shortest path $P_*$ in $G$ from $s_*$ to $t_*$ has minimum
    length. If no such path exists halt and output $\mathcal{A}$.
Step 3. Add $P_*$ to $\mathcal{A}$ and remove the edges of $P_*$ from $G$. Remove $(s_*, t_*)$ from $\mathcal{T}$. Goto Step 2.
```

We begin by giving an upper bound on the approximation ratio of the algorithm.

Theorem 4.1 *Algorithm* GREEDY_PATH *runs in polynomial time and outputs a solution to an edge-disjoint paths problem $(G = (V, E), \mathcal{T})$ of size at least $1/(\sqrt{|E_o|} + 1)$ times the optimum, where $E_o \subseteq E$ is the set of edges used by the paths in an optimal solution.*

We now prove improved bounds on the size of the realizable set output by GREEDY_PATH.

Theorem 4.2 *Algorithm* GREEDY_PATH *outputs a solution to an edge-disjoint path problem $(G = (V, E), \mathcal{T})$ of size $\Omega(OPT^2/|E_o|)$, where $E_o \subseteq E$ is the set of edges used by the paths in an optimal solution.*

Proof. Let t be the total number of iterations of GREEDY_PATH and $\mathcal{A}_i$ be the set $\mathcal{A}$ at the end of the i-th iteration. Let $\mathcal{O}$ be an optimal set of paths. We say that a path P_x *hits* a path P_y if P_x and P_y share an edge. We define the set $\mathcal{O} \ominus \mathcal{A}$ as the paths in $\mathcal{O}$ that correspond to commodities not routed in $\mathcal{A}$. Let P_i be the path added to $\mathcal{A}$ at the i-th iteration of the algorithm. If P_i hits k_i paths in $\mathcal{O} \ominus \mathcal{A}_i$ that are not hit by a path in $\mathcal{A}_{i-1}$, then P_i must have length at least k_i. In turn each of the paths hit has length at least k_i otherwise it would have been selected by the algorithm instead of P_i. Furthermore all paths in $\mathcal{O}$ are edge-disjoint with total number of edges $|E_o|$. Therefore $\sum_{i=1}^{t} k_i^2 \leq |E_o|$. Applying the Cauchy-Schwartz inequality on the left-hand side we obtain that $(\sum_{i=1}^{t} k_i)^2/t \leq |E_o|$. But $\sum_{i=1}^{t} k_i = |\mathcal{O} \ominus \mathcal{A}_t|$ since upon termination of the algorithm all paths in $\mathcal{O} \ominus \mathcal{A}_t$ must be hit by some path in $\mathcal{A}_t$. We obtain $\frac{|\mathcal{O} \ominus \mathcal{A}_t|^2}{t} \leq |E_o|$. Without loss of generality we can assume that $|\mathcal{A}_t| = o(|\mathcal{O}|)$, since otherwise GREEDY_PATH obtains a constant-factor approximation. It follows that $t = \Omega(|\mathcal{O}|^2)/|E_o|) = \Omega(OPT^2/|E|)$. ∎

Corollary 4.1 *Algorithm* GREEDY_PATH *outputs a solution to an edge-disjoint path problem $(G = (V, E), \mathcal{T})$ of size $\Omega(OPT/d_o)$, where d_o is the average length of the paths in an optimal solution.*

Algorithm GREEDY_PATH gives a solution to vertex-disjoint paths with the following modification at Step 3: remove the vertices of P_* from G. Call the resulting algorithm GREEDY_VPATH. The analogues of the results above can be seen to hold for GREEDY_VPATH as well.

Theorem 4.3 *Algorithm* GREEDY_VPATH *outputs a solution to a vertex-disjoint path problem $(G = (V, E), \mathcal{T})$ of size $\Omega(\max\{OPT/|V_o|, OPT^2/|V_o|, OPT/d_o\})$, where $V_o \subseteq V$ is the set of vertices used by the paths in an optimal solution and d_o is the average length of the paths in an optimal solution.*

Acknowledgments. We wish to thank Jon Kleinberg and Aravind Srinivasan for valuable discussions. We also thank Javed Aslam for helpful discussions.

References

[1] R. Aharoni, P. Erdös, and N. Linial. Optima of dual integer linear programs. *Combinatorica*, 8:13–20, 1988.

[2] Y. Aumann and Y. Rabani. Improved bounds for all-optical routing. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 567–576, 1995.

[3] A. Z. Broder, A. M. Frieze, and E. Upfal. Static and dynamic path selection on expander graphs: a random walk approach. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 531–539, 1997.

[4] C. Cooper. The thresold of hamilton cycles in the square of a random graph. *Random Structures and Algorithms*, 5:25–31, 1994.

[5] H. Fleischner. The square of every two-connected graph is hamiltonian. *Journal of Combinatorial Theory B*, 16:29–34, 1974.

[6] A. Frank. Packing paths, cuts and circuits - a survey. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout*, pages 49–100. Springer-Verlag, Berlin, 1990.

[7] N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18:3–20, 1997.

[8] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. V. Vazirani, and V. V. Vazirani. Global wire routing in two-dimensional arrays. *Algorithmica*, 2:113–129, 1987.

[9] R.M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

[10] J. M. Kleinberg. *Approximation algorithms for disjoint paths problems.* PhD thesis, MIT, Cambridge, MA, May 1996.

[11] J. M. Kleinberg. Single-source unsplittable flow. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 68–77, October 1996.

[12] J. M. Kleinberg and R. Rubinfeld. Short paths in expander graphs. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 86–95, 1996.

[13] J. M. Kleinberg and É. Tardos. Disjoint paths in densely-embedded graphs. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 52–61, 1995.

[14] S. G. Kolliopoulos and C. Stein. Improved approximation algorithms for unsplittable flow problems. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 426–435, 1997.

[15] F. T. Leighton and S. B. Rao. Circuit switching: a multi-commodity flow approach. In *Workshop on Randomized Parallel Computing*, 1996.

[16] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 422–431, 1988.

[17] Y.-L. Lin and S. E. Skiena. Algorithms for square roots of graphs. *SIAM Journal on Discrete Mathematics*, 8(1):99–118, February 1995.

[18] L. Lovász. On the ratio of optimal and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[19] P. Martin and D. B. Shmoys. A new approach to computing optimal schedules for the job-shop scheduling problem. In *Proceedings of the 5th Conference on Integer Programming and Combinatorial Optimization*, pages 389–403, 1996.

[20] D. Peleg and E. Upfal. Disjoint paths on expander graphs. *Combinatorica*, 9:289–313, 1989.

[21] S. Plotkin. Competitive routing of virtual circuits in ATM networks. *IEEE J. Selected Areas in Comm.*, pages 1128–1136, 1995. Special issue on Advances in the Fundamentals of Networking.

[22] S. Plotkin, D. B. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[23] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.

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

[25] N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout.* Springer-Verlag, Berlin, 1990.

[26] A. Schrijver. Homotopic routing methods. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout.* Springer-Verlag, Berlin, 1990.

[27] D. B. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. *SIAM Journal on Computing*, 23(3):617–632, June 1994.

[28] A. Srinivasan. Improved approximations of packing and covering problems. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 268–276, 1995.

[29] A. Srinivasan. An extension of the Lovász Local Lemma and its applications to integer programming. In *Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 6–15, 1996.

[30] A. Srinivasan. Improved approximations for edge-disjoint paths, unsplittable flow and related routing problems. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 416–425, 1997.

[31] A. Srinivasan and C.-P. Teo. A constant-factor approximation algorithm for packet routing and balancing local vs. global criteria. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 636–643, 1997.

[32] C. Stein. *Approximation algorithms for multicommodity flow and shop scheduling problems.* PhD thesis, MIT, Cambridge, MA, August 1992. Also appears as MIT/LCS/TR-550.

[33] P. Underground. On graphs with hamiltonian squares. *Discrete Mathematics*, 21:323, 1978.

Appendix

A different bound for column-restricted PIP's. We show how one can modify the first two steps of algorithm COLUMN_PARTITION to obtain a new approximation bound in addition to those stated in Theorem 2.2.

Let δ_2, δ_3 be constants in $(0,1)$ such that $\zeta\delta_3 = \delta(1-\delta_2)$. Note that given a $\delta < 1$, it is always possible to select δ_2, δ_3 so that $\zeta \geq 1$ and vice versa. We partition the interval $(0,1]$ of demands into ξ geometrically increasing subintervals $(0, \delta_3(\delta_2)^{\xi-2}], \ldots, (\delta_3(\delta_2)^{i+1}, \delta_3(\delta_2)^i], \ldots, (\delta_3\delta_2, \delta_3], (\delta_3, 1]$, such that $\delta_3(\delta_2)^{\xi-2} \leq 1/n^k$ for suitably large constant $k > 1$. We now run COLUMN_PARTITION from Step 3 and on. Each of the three vectors $\dot{x}, \hat{x}, \bar{x}$ solves a packing problem with demands lying in $(0, 1/n^k]$, $(1/n^k, \delta_3]$ and $(\delta_3, 1]$ respectively. Let $\dot{\mathcal{P}}, \hat{\mathcal{P}}, \bar{\mathcal{P}}$ be the three induced packing problems. The optimal solution to the linear relaxation of at least one of them will have value at least $1/3$ of the optimal solution to the linear relaxation of $\mathcal{P}$. It remains to lower bound the approximation achieved by each of the three vectors on its corresponding domain of demands. Since m, B, ζ and α are fixed for all subproblems, note that σ is a function of one variable, y^*.

Vector $\dot{g}$ solves $\dot{\mathcal{P}}$ optimally. The solution is feasible since all demands are less than $1/n^k$ and thus the value of the left-hand side of any packing constraint cannot exceed 1. By Lemma 2.1 vector $\bar{g}$ outputs a solution to $\bar{\mathcal{P}}$ of value at least $\sigma(m, B, \alpha, \delta_3 y_*^{\delta_3, 1})$.

Deriving the analogue of Lemma 2.3 one can establish for $\hat{\mathcal{P}}$ that the value of the solution output is given by the sum $A = \sum_{i=1}^{\xi-2} \sigma(m, \zeta B, \alpha, \delta_2(1-\delta) y_*^{\delta_3(\delta_2)^i, \delta_3(\delta_2)^{i-1}})$. If function σ is convex the sum above is minimized when all the terms $y_*^{\delta_3(\delta_2)^i, \delta_3(\delta_2)^{i-1}}$, $1 \leq i \leq \xi - 2$, are equal to $y_*^{n^{-k}, \delta_3}/(k' \log n)$ where k' is a constant chosen so that $k' \log n$ is equal to $\xi - 2$. Instantiating $\sigma(y_*)$ with $\Omega(y_*(y_*/m)^{1/B})$ we obtain the following theorem.

Theorem 4.4 *Let $\mathcal{P} = (A, b, c)$ be a column-restricted PIP. Let y^* be the optimum of the linear relaxation of $\mathcal{P}$ and ζ be a constant greater than or equal to 1. There is a polynomial time algorithm that outputs a solution to $\mathcal{P}$ of value*

$$T(m, \zeta B, y^*) = \Omega\left(\max\left\{ y_*^{0, n^{-k}}, y_*^{n^{-k}, \delta_3}\left(\frac{y_*^{n^{-k}, \delta_3}}{m \log n}\right)^{1/(\zeta B)}, y_*^{\delta_3, 1}\left(\frac{y_*^{\delta_3, 1}}{m}\right)^{1/B}\right\}\right).$$

Let us comment on the above bound. Comparing the function $y^*(y^*/m \log n)^{1/(\zeta B)}$ to the function $y^*(y^*/m)^{1/(B)}$ there is a $\log n$ in the denominator. However the exponent is $1/(\zeta B)$ for *any* constant ζ which makes this bound particularly powerful and in fact $\omega(y^*(y^*/m)^{1/(B)})$ in the case when the middle term dominates and $y_*^{n^{-k}, \delta_3} < m \log n$.

Proof of Theorem 3.2. The algorithm is similar to MAX_ROUTING but omits the Steps 2 and 3 that find the single-path fractional routing. Step 4 is modified as follows. Let $1, \ldots, M$ be an arbitrary ordering of the paths in the flow decomposition of f. A has $M + |T|$ total columns, one for each path in the decomposition and an additional $|T|$ special columns. The number of rows is $|E| + |T|$. The first $|E|$ rows correspond to capacity constraints in G and their entries are filled in the manner described in Algorithm MAX_ROUTING. The remaining rows correspond to constraints that enforce that only one path is chosen per commodity in the final integral solution. In particular row $|E| + i$, $1 \leq i \leq |T|$, has ρ_i in column j, $1 \leq j \leq M$, if the j-th path in the flow decomposition carries flow for commodity i; otherwise the entry is zero. The entries of column $M + l$, $1 \leq l \leq |T|$, are all zero except for the $(|E| + l)$-th row where the entry is equal to $1 - \rho_l$. Finally each entry $b_{|E|+i}$, $1 \leq i \leq |T|$, of the b-vector is set to 1. We set c_i, $1 \leq i \leq M$ to the weight of the commodity that flows along the i-th path in the decomposition. The entries in the c-vector past the M-th position are set to 0.

It is straightforward to verify that the above construction enforces an integral solution where at most one of the fractional paths is chosen for each commodity. The resulting PIP is column-restricted with fractional optimum y^* and maximum number of non-zero entries in a column equal to $d+1$. The theorem follows.