Stochastic Unsplittable Flows*

Anupam Gupta[1] and Archit Karandikar[2]

1 Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

2 Facebook, Inc., Menlo Park, CA, USA

Abstract

We consider the stochastic unsplittable flow problem: given a graph with edge-capacities, k source-sink pairs with each pair $\{s_j, t_j\}$ having a size S_j and value v_j, the goal is to route the pairs unsplittably while respecting edge capacities to maximize the total value of the routed pairs. However, the size S_j is a random variable and is revealed only after we decide to route pair j. Which pairs should we route, along which paths, and in what order so as to maximize the expected value?

We present results for several cases of the problem under the no-bottleneck assumption. We show a logarithmic approximation algorithm for the single-sink problem on general graphs, considerably improving on the prior results of Chawla and Roughgarden which worked for planar graphs. We present an approximation to the stochastic unsplittable flow problem on directed acyclic graphs, within less than a logarithmic factor of the best known approximation in the non-stochastic setting. We present a non-adaptive strategy on trees that is within a constant factor of the best adaptive strategy, asymptotically matching the best results for the non-stochastic unsplittable flow problem on trees. Finally, we give results for the stochastic unsplittable flow problem on general graphs.

Our techniques include using edge-confluent flows for the single-sink problem in order to control the interaction between flow-paths, and a reduction from general scheduling policies to "safe" ones (i.e., those guaranteeing no capacity violations), which may be of broader interest.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Approximation Algorithms, Stochastic optimization, confluent flows, unsplittable flows

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2017.7

1 Introduction

We consider the following stochastic problem of routing uncertain demands in a network. We are given a graph $G = (V, E)$ with edge capacities c_e and a set J of k source-sink pairs $\{s_j, t_j\}$ (called *jobs*). We want to route some flow from each source to its corresponding sink, but the amount of flow to be sent for job j (called its *size*) is not known *a priori*. We only know that its size is a random variable S_j, with a known distribution. (We assume that the sizes of jobs are independent of each other.) Each job has a value v_j. We will operate under the prevalent *no-bottleneck assumption* (NBA). In our setting, this means that the maximum size in the support of any job's distribution is at most the minimum capacity of any edge in the graph.

* Research partly supported by NSF awards CCF-1319811, CCF-1540541 and CCF-1617790. Work done when A. Karandikar was at the Computer Science Department, Carnegie Mellon University.

We want a routing strategy that decides on jobs to route in the network. This involves us repeatedly choosing an uninstantiated job j and a path P_j for it, and routing this job along the path. Once we do this, the size S_j is instantiated, drawn according to the given probability distribution. If S_j is at most the residual capacity (which initially equals capacity) on each edge of path P_j, the routing is considered successful, we get its value v_j and the residual capacity of all edges in P_j reduces by S_j. Else if there is some edge $e \in P_j$ with residual capacity less than S_j, the routing is unsuccessful and we do not get its value. Moreover, each such "violated" edge is henceforth considered "forbidden" and cannot be used on subsequent paths. When a job j is routed unsuccessfully on a path, it still uses up capacity S_j on all edges along that path that do not become forbidden. The goal is to find a strategy that maximizes the expected cumulative value of jobs it routes successfully. This problem is the stochastic version of the well-known *unsplittable flow problem* (UFP), and as such, we call it the *stochastic unsplittable flow problem* (sUFP).

A strategy for routing jobs is allowed to be *adaptive*, i.e., it can use results of its previous decisions to make its current decision. In contrast *non-adaptive* policies provide a sequence of jobs to route upfront. Storing as well as finding adaptive policies can potentially be exponential in the size of the input and so finding non-adaptive policies of expected value close to the optimal expected value of an adaptive strategy is desirable. The *adaptivity gap* is the ratio of the value of the optimal adaptive strategy to the optimal non-adaptive strategy.

The *stochastic knapsack* problem was first studied by Dean, Goemans and Vondrak [15] who showed that it has a constant adaptivity gap. The stochastic knapsack is the special case of the sUFP on a graph with a single edge. Subsequent work of Dean et al. [14] also considered several versions of the stochastic packing problem, which is a generalization of the sUFP. Over a universe of size d, they showed an $O(\sqrt{d})$ adaptivity gap for stochastic set-packing, and $O(d)$ for general packing problems. Bansal et al. [2] gave an $O(k)$-adaptivity gap for stochastic set-packing with sets of size at most k.

The sUFP was first studied by Chawla and Roughgarden [8]. They studied the *single-sink stochastic routing* problem (SSSR), where all the targets t_i are the same vertex t, and assumed the stronger α-NBA, i.e., the size of each job is supported on $[0, \alpha c_{\min}]$ for some $\alpha < 1$. For planar graphs, they presented a logarithmic approximation algorithm which guaranteed no capacity violations. This work left open several interesting directions: can we work under the NBA instead of the stronger α-NBA? Can we go beyond planar graphs to handle general graphs? How about going beyond single-sink instances, and giving results for more general unsplittable flow instances?

In this paper, we initiate a broader study of the sUFP, and answer these questions in the positive. As is common in the existing research on their deterministic versions, we assume that the underlying graph G is undirected for purposes of the sUFP on trees and general graphs, and that G is a digraph in our treatment of the SSSR and the sUFP on DAGs.

1.1 Our Results

Single-Sink Stochastic Routing Problem (SSSR). Our main result is for the SSSR; as defined above, here all the sinks are co-located. For this result, define the *weight* of job j as $w_j := v_j/\mu_j$, where v_j is the value and $\mu_j = \mathbf{E}[S_j]$ is the expected size of the job.

▶ **Theorem 1.1** (SSSR). *The single-sink stochastic routing problem (under the no-bottleneck assumption) has a poly-time $O(\min(\log k, \log W))$-approximation algorithm. Here k is the number of jobs in the instance, and $W := \frac{\max_j w_j}{\min_j w_j}$ is the maximum ratio between the weights of the jobs.*

Chawla and Roughgarden [8] showed a *safe* $O(\frac{\log W}{1-\alpha})$-approximation for *planar* SSSR instances under the α-NBA; here, safe means the policy is guaranteed to have no edge-capacity violations. No results prior to ours were known for the SSSR on general graphs. In fact, we can also extend Theorem 1.1 to show that under the α-NBA, we get a *safe* $O(\frac{\min(\log k, \log W)}{1-\alpha})$ approximation for the SSSR on general graphs. Recall that for the non-stochastic version of this problem, Dinitz et al. [16] gave a constant-factor approximation algorithm. To obtain the aforementioned logarithmic approximation, we will show a simple general reduction from edge-confluence to node-confluence that was proposed as an open direction by Chen et al. [13].

Stochastic Unsplittable Flow Problem (sUFP) on Directed Acyclic Graphs. Chekuri et al. [10] gave an $O(\sqrt{n})$ approximation for the UFP. We obtain an analogous result in the stochastic setting, giving away further a factor of $O(\sqrt{\log k})$. Recall that k is the number of jobs.

▶ **Theorem 1.2** (sUFP on DAGs). *The stochastic unsplittable flow problem (under the no-bottleneck assumption) has a poly-time $O(\sqrt{n \log k})$-approximation algorithm on directed acyclic graphs.*

Stochastic Unsplittable Flow Problem (sUFP) on Trees. Our next result is for the sUFP on trees. Here, the s_j-t_j paths are unique, which means the routing strategy merely has to decide the sequence of jobs to route.

▶ **Theorem 1.3** (sUFP on Trees). *The stochastic unsplittable flow problem on trees (under the no-bottleneck assumption) has a non-adaptive poly-time $O(1)$-approximation algorithm.*

To the best of our knowledge, this is the first result for the sUFP on trees. Our result follows as a corollary to a more general result, where each job corresponds to a "spider"; we present this result in the full version of this paper. The non-stochastic unsplittable flow problem on trees admits a constant factor approximation under the NBA, by a result of Chekuri et al. [11], and hence our result extends this to the stochastic realm.

Stochastic Unsplittable Flow Problem (sUFP) on General Graphs. For UFP on general graphs, Chakrabarti et al. [7] gave an $O(F_G \log n) = O(\Delta\alpha^{-1} \log^2 n)$ approximation for the UFP. Here F_G denotes the *flow number*, α denotes the expansion and Δ denotes the maximum degree of the graph. (These quantities will be formally defined in Section 4.) Our next result shows how to match these approximation guarantees in the stochastic setting. The proof is contained in the full version of the paper.

▶ **Theorem 1.4** (sUFP). *The stochastic unsplittable flow problem (under the no-bottleneck assumption) has a non-adaptive poly-time $O(d)$-approximation algorithm if the LP relaxation sends flows along paths of length at most d. This can be extended to an $O(F_G \log n) = O(\Delta\alpha^{-1} \log^2 n)$-approximation algorithm on general graphs.*

Safe Routing Strategies. Finally, we give an approach to convert a general strategy (under the NBA) to a safe one (assuming the α-NBA). For the sUFP on general graphs, directed acyclic graphs and trees, we can convert policies under the NBA to safe policies under the α-NBA by sacrificing a factor of $O(\frac{1}{1-\alpha})$ for $\alpha \in (0, \frac{1}{2}]$. For stochastic knapsack and the SSSR such a transformation can be performed for all $\alpha \in (0, 1)$.

1.1.1 Our Techniques

A primary question when dealing with stochastic problems is this: how can we argue about the optimal strategy, which is given by an (exponential-sized) decision-tree? One appealing approach – which we employ here – is to write an "average" LP relaxation which tries to send the average amount of flow for each job. A feasible solution to this linear program is to set the variables for job j based on the probability that the optimal strategy routes j, and hence the LP value gives us an upper bound on the optimal value. However, for stochastic problems, it is not enough to round the solution to integers: indeed, an integer solution to this LP does not directly give us a good strategy (the constraints suffice only when each job behaves like its expectation). Hence we need to "interpret" this solution to get a feasible strategy.

For example, in the SSSR, suppose we are given unsplittable flows that send $\mu_j = \mathbf{E}[S_j]$ amount of flow from s_j to the sink t, for every job j. We may hope to say that each job can be routed with constant probability. However, the flow-paths can interfere in complicated ways, and it is difficult to lower bound the probability that there is "enough room" for some job deep in the process. Our new idea is to alter the flow paths to make them *confluent* – i.e., when two flows use a common edge, they flow together from that point on to the sink. The logarithmic losses come from this step. The confluent flows now behave in a tree-like fashion, and the bottleneck edges are now those incident to the root. We can then argue that these edges are not over-congested with reasonable probability.

For the sUFP in directed acyclic graphs we crucially use our confluence techniques along with idea of v-separation inspired by Chekuri et al for rounding "small" jobs on long flow paths. The other cases are handled using the rounding techniques mentioned above.

To translate arbitrary policies on general graphs to safe policies on unit-capacity graphs, we show how to transform the given set of jobs into new jobs with the same expected size but truncated job sizes, on which we can run the general non-safe strategy. The saved space can then be used to ensure that our real jobs never run out of space.

The sUFP on paths and trees is a natural extension of the well-studied stochastic knapsack problem, and can be viewed as a set of spatially-correlated knapsack problems. Here, we show that for jobs with "large" expected size, we can get good value (comparable to the LP value) by routing an essentially "disjoint" set of jobs. Jobs with small expected size are routed using a scaled-down version of their LP variables. We can then go over the jobs in a certain order, and show that each job, if routed, has a constant probability of having enough capacity to be able to successfully route. A similar plan works for Theorem 1.4 for the sUFP on general graphs: the union bound is over the d edges, and we lose an $O(d)$ term. The translation to the flow number and expansion is standard. The proofs for our results for the sUFP on paths and trees and on general graphs can be found in the full version of this paper.

1.2 Related Work

After the pioneering work of [15], improved algorithms for the stochastic knapsack problem were given by Bhalgat et al. [4, 3], by combining bi-criteria adaptive strategies (another bi-criteria algorithm was given by Li and Yuan [21]), and an LP-rounding approach; we do not know how to implement such adaptive strategies in our case. Work on stochastic knapsack was extended to multi-armed bandits (see, e.g., [17, 18]), and correlated rewards and sizes [19, 21, 22]; all these write more sophisticated LPs to capture correlations. Extending our routing/packing problems to these correlated settings seems non-trivial, and remains an exciting direction for future work. The only prior work on stochastic routing is that of Chawla and Roughgarden [8] discussed above.

Our work on sUFP on trees is directly inspired by work by Chakrabarti et al. [6, 7] and Chekuri et al. [11] on resource-allocation problems and unsplittable flow on paths and trees. These papers get better approximations by combining LP rounding approaches with dynamic programming (DP) for large item sizes, but extending the DP approach to the stochastic case seems difficult. Some variants of the sUFP on Trees (e.g., equal edge-capacities, packing subtrees) are given in the Master's thesis of the second-named author [20]. Algorithms removing the NBA also rely heavily on dynamic programming (see [5, 1] and references), though the LP-based approaches of Chekuri et al. [9] offer hope as well. The unsplittable flow problem, both on general graphs and on trees/paths has been widely studied; see, e.g., the references in [9]. Our results for the sUFP on directed acyclic graphs are based on the work by Chekuri et al [10].

For the single-sink routing problem, we are unable to directly extend the constant-factor approximation of Dinitz et al. [16] to the stochastic case. Instead we use ideas based on confluent flows, which were first developed by Chen et al. [13, 12]. In very recent work, Shepherd, Vetta, and Wilfong [24] showed that for general capacitated networks, under the NBA, there is a $O(\log^6 n)$-approximation algorithm for the demand maximization problem. Shepherd and Vetta [23] give hardness results for such problems.

2 Additional Notation

Here we recall some essential notation introduced in §1 and introduce some new notation. An instance of sUFP consists of a set J containing k jobs, each having a source-sink pair $\{s_j, t_j\}$, a value v_j, and random size S_j. We assume that the distribution of the r.v. S_j is known to us; most of our algorithms only require knowing the mean μ_j. Each edge e of the given graph $G = (V, E)$ has a capacity c_e. Let $c_{\min} := \min_e c_e$ be the minimum capacity of any edge, and $D_{\max} := \min\{d \mid \Pr[S_j > d] = 0 \;\; \forall j \in J\}$. The *no-bottleneck assumption (*NBA*)* requires that $D_{\max} \le c_{\min}$. By scaling we will always imagine that $c_{\min} = 1$, hence under the NBA, we can assume that $\max_e c_e \le k$, where k is the total number of jobs. If the sizes are deterministic, we call the problem the *unsplittable flow problem* (UFP); the goal is to route the maximum value set of jobs while respecting edge-capacities. If all sizes are deterministic and equal, we get the *capacitated* EDP (edge-disjoint paths) problem. (In this case we assume that all the jobs are unit-sized, and all the capacities are integers.)

2.1 An LP Relaxation

Given edge capacities $c_e \in \mathbb{R}_{\ge 0}$, and a set J of jobs with demands μ_j, we upper-bound the expected value of the optimal adaptive strategy using the following multicommodity-flow linear program $LP_{UFP}(J, \mathbf{c})$:

$$\begin{aligned} \phi(J,\mathbf{c}) := \qquad & \max \textstyle\sum_j (v_j/\mu_j) x_j & & (LP_{UFP}) \\ & x_j \le \mu_j & & \forall j \in J \\ & x_j = \textstyle\sum_{P \in \mathcal{P}(s_j,t_j)} f_P & & \forall j \in J \\ & \textstyle\sum_{P : e \in P} f_P \le c_e & & \forall e \in E \\ & f_P \ge 0 & & \forall P \end{aligned}$$

Here $\mathcal{P}(u, v)$ is the set of all paths from vertex u to v. The same linear program is valid for both directed and undirected instances, with the definition of $\mathcal{P}$ varying between the two. The following theorem is analogous to a result of [15] for stochastic knapsack, and has been used previously [8, 7, 14].

▶ **Theorem 2.1.** *The value of the optimal adaptive strategy for a stochastic routing problem with edge capacity vector* $\mathbf{c}$ *(under the* NBA*, where* $D_{\max} \le c_{\min} = 1$*) and a set* J *of jobs with expected sizes* $\boldsymbol{\mu}$ *is at most* $\phi(J, \mathbf{c} + \mathbf{1})$*. Using* NBA *and scaling, we get* $\phi(J, \mathbf{c} + \mathbf{1}) \le \phi(J, 2\mathbf{c}) \le 2\,\phi(J, \mathbf{c})$*.*

3 Single-Sink Stochastic Routing

We now give a logarithmic approximation for the single-sink stochastic routing (SSSR) problem (under the NBA) on general directed graphs. This improves on the logarithmic guarantee given by Chawla and Roughgarden for planar instances. To understand why this problem is not just the single-sink UFP problem, suppose we are given a routing sending the μ_j flow from each source unsplittably to the sink. To solve the stochastic problem, we have to account for the randomness in the sizes – if we route P_1 and it takes on size greater than its expectation μ_j, what should we do next?

Our main insight is the use of *edge-confluent flows*, which may be somewhat unexpected but is natural in hindsight. A flow in a single-sink network is *confluent* if any two flows which "meet" are merged from there onwards. (One can have edge-confluent or node-confluent flows; the formal definitions appear below.) To get a high-level idea, observe that if we solve the relaxation (LP_{UFP}), and the flow happens to be edge-confluent, the interference between flow-paths can be controlled by controlling the interference on the edges incoming into the sink t.

Our approach is the following: we convert the non-confluent solution to (LP_{UFP}) to an edge-confluent flow. This is not immediate: existing results deal with node-confluence and are applicable only for unit-capacity networks, whereas our SSSR instances have general capacities. Next, we reduce this edge-confluent flow to several instances of the stochastic knapsack problem, one corresponding to every edge incoming to the sink in the unit-capacity network. Our algorithm is adaptive, but only "mildly" so: the adaptivity arises only from the preemption of jobs in each stochastic knapsack instance in order to keep the used-capacity within control. We use the NBA during the conversion to edge-confluent flows. Under the stronger α-NBA, the algorithm is safe.

3.1 Confluent Flows

Given a *directed* graph $G = (V, E)$ with a special sink vertex t, and a set of sources $S = \{s_1, s_2, \ldots, s_k\} \subseteq V$, a *node-confluent flow* is a flow from the sources to the sink such that for each non-sink vertex $v \in V$, all the flow exiting v uses a single arc leaving v. An *edge-confluent* flow is one where for each arc $e \in E$, all flow using this edge must subsequently share the same arcs in their journey to the sink. Equivalently, for an edge-confluent flow $\mathbf{f}$, there exists a mapping $\phi_v : E \to E$ that maps for each vertex v the in-arcs I_v of v to its out-arcs O_v, such that for each out-arc $e = (v, w) \in O_v$, $f_e = \sum_{e' \in I_v : \phi_v(e') = e} f_{e'}$.

For our edge-confluence results, we will operate in a setting where all edges have unit capacity. In this context, we define the the congestion of a flow $\mathbf{f}$ as $\mathsf{cong}(\mathbf{f}) = \max\{1, \max_{e \in E} f_e\}$. We denote the total amount of flow reaching sink t by $|\mathbf{f}|$. The results of Chen et al. [12] for node-confluence can be transferred to edge-confluence to get the following result.

▶ **Theorem 3.1.** *Consider a directed single-sink flow network with* <u>*unit*</u> *edge-capacities under the* NBA*, and a flow* $\mathbf{f}$ *sending* d_i *units of flow from source* s_i *to the sink, respecting edge-capacities. (I.e.,* $\mathsf{cong}(\mathbf{f}) \le 1$*.) Then the following exist and can be found in polynomial time.*

1. *An edge-confluent flow* $\mathbf{f}'$ *that for each* $i \in [k]$ *sends* d_i *flow from* s_i *to* t *(i.e.,* $|\mathbf{f}'| = |\mathbf{f}|$*) so that*
$$\mathsf{cong}(\mathbf{f}') \leq 1 + \ln k \qquad \text{and} \qquad |\mathbf{f}'| = |\mathbf{f}|\,.$$
2. *A subset* $R \subseteq S$ *and an edge-confluent flow* $\mathbf{f}''$ *which for each* $i \in R$ *sends* d_i *flow from* s_i *to* t *such that* $\sum_{i\in R} d_i \geq \frac{1}{3}\sum_{i\in[k]} d_i$*, and so that*
$$\mathsf{cong}(\mathbf{f}'') = 1 \qquad \text{and} \qquad |\mathbf{f}''| \geq \frac{|\mathbf{f}|}{3}\,.$$

Hence, the first result presents a way to route all the jobs while incurring logarithmic congestion, and the second result presents a way to route a large subset of the jobs and incur unit congestion.

Proof Sketch. The idea is to construct the line graph H of the given digraph G (plus one extra node) so that node-confluent flows in the given network correspond to edge-confluent flows in the constructed network. (See Figure 1.) Now given a fractional flow in G, we can map this flow to H, use a result of Chen et al. on transforming general flows to node-confluent flows in H, and transform the resulting node-confluent flow back to an edge-confluent flow in G. The formal proof appears in Appendix A.2. ◀

▶ **Corollary 3.2.** *Consider a directed single-sink flow network with* <u>*unit*</u> *edge-capacities under the* NBA*, and a flow* $\mathbf{f}$ *sending* d_i *units of flow from source* s_i *to the sink, respecting edge-capacities. (I.e.,* $\mathsf{cong}(\mathbf{f}) \leq 1$*.) Moreover, each source* s_i *has weight* w_i*, and let* $\mathbf{w}$ *denote the vector of weights.*

Then we can find, in polynomial time, an edge-confluent *flow* $\widehat{\mathbf{f}}$ *sending* $\widehat{d_i}$ *units of flow from* s_i *to the sink respecting edge-capacities (i.e.,* $\mathsf{cong}(\widehat{\mathbf{f}}) = 1$*), such that each* $\widehat{d_i} \in [0, d_i]$*,*
$$\langle \mathbf{w}, \widehat{\mathbf{d}}\rangle \geq \langle \mathbf{w}, \mathbf{d}\rangle \cdot \frac{1}{\min\{1 + \ln k, 6(1 + \log_2 W)\}},$$
where $W := \frac{\max_j w_j}{\min_j w_j}$.

Proof. We want to find a flow $\widehat{\mathbf{f}}$ for which $\langle \mathbf{w}, \widehat{\mathbf{d}}\rangle$ is within a logarithmic factor of $\langle \mathbf{w}, \mathbf{d}\rangle = \sum_{i=1}^{k} w_i d_i$. Apply Theorem 3.1(1) to the flow $\mathbf{f}$ to obtain edge-confluent flow $\mathbf{f}'$. Scaling the flow $\mathbf{f}'$ down by a factor of $1 + \ln k$ gives us a flow $\widehat{\mathbf{f}}$ with $\langle \mathbf{w}, \widehat{\mathbf{d}}\rangle = \sum_{i\in[k]} w_i \cdot \widehat{d_i} \geq \frac{\langle \mathbf{w},\mathbf{d}\rangle}{1+\ln k}$.

Next, bucket the weights w_i into dyadic intervals. By averaging, there exists some interval $I = (2^j, 2^{j+1}]$ such that jobs with weights in this interval have $\sum_{i:w_i\in I} w_i \cdot d_i \geq \frac{\langle \mathbf{w},\mathbf{d}\rangle}{1+\log_2 W}$. Use Theorem 3.1(2) to get $R \subseteq \{i : w_i \in I\}$ and flow $\mathbf{f}''$ that sends flow $|\mathbf{f}''| = \sum_{i\in R} d_i \geq \frac{1}{3}\sum_{i:w_i\in I} d_i$. Since the weights of jobs in I are within a factor of 2 of each other, we get that $\sum_{i\in R} w_i \cdot d_i \geq \frac{1}{6}\sum_{i:w_i\in I} w_i \cdot d_i \geq \frac{\langle \mathbf{w},\mathbf{d}\rangle}{6(1+\log_2 W)}$. The better of these two edge-confluent flows gives us the claim. ◀

3.2 Approximate Single-Sink Stochastic Routing using Confluent Flows

Consider an instance of the SSSR on the directed graph $G = (V, E)$ under the no-bottleneck assumption (NBA) scaled so that $c_{\min} = 1$. Assume that each source s_i is a unique vertex in G with a single out-edge of capacity 1. This assumption is without loss of generality, since we can always create a new vertex for each source and attach it using a unit-capacity edge to the old location. This does not change feasibility because of the NBA.

For each edge e, define $K_e = \lceil \lfloor c_e \rfloor / 2 \rceil$. Note that $K_e > c_e/3$ for all e. Given such a digraph $G = (V, E)$, define a (multi)graph $G' = (V, E')$ where for each edge $e = (u, v) \in E(G)$, we have K_e parallel unit-capacity edges (u, v) in E'.

The following corollary states a way to obtain a confluent solution to LP_{UFP} for G' within a logarithmic factor of the optimal solution to LP_{UFP} for G. Define the weight w_j for source S_j as v_j/μ_j. Before we state it, recall the definition of LP_{UFP} for G from §2.1 and note that an optimal solution $(\mathbf{x}, \mathbf{f})$ to it is a flow $\mathbf{f}$ in the graph G such that the total weight of this flow $\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{j \in J} w_j x_j = \phi(J, \mathbf{c})$.

▶ **Corollary 3.3.** *Given a solution $(\mathbf{x}, \mathbf{f})$ to LP_{UFP} for the* SSSR *instance on the graph G, there exists an edge-confluent solution $(\mathbf{x}', \mathbf{f}')$ to LP_{UFP} on the unit-capacity graph G', such that*

$$\langle \mathbf{w}, \mathbf{x}' \rangle \geq \langle \mathbf{w}, \mathbf{x} \rangle \cdot \Omega\left(\frac{1}{\min\{\log k, \log W\}} \right),$$

where the weight of job s_j is $w_j = v_j/\mu_j$, and $W := \frac{\max_j w_j}{\min_j w_j}$. Moreover, all the $x'_j \leq \mu_j$ units of flow from s_j to the sink is unsplittably routed. Finally, this flow can be found in time $\mathrm{poly}(n, k)$.

Proof. The solution $(\mathbf{x}, \mathbf{f})$ on G can be ported to a solution $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{f}})$ on G' via scaling down by at most a factor of 3, since the parallel edges replacing each edge e have total capacity $K_e > c_e/3$. Now apply Corollary 3.2 with $d_j = \widetilde{x}_j$, and $w_j = (v_j/\mu_j)$ to get an edge-confluent flow $(\mathbf{x}', \mathbf{f}')$ with the claimed value.

Moreover, since each source has a single out-edge, all flow from s_j to the sink in G' must use this edge. Now edge-confluence ensures that this flow must be routed unsplittably to the sink.

To bound the run-time, observe that the NBA implies that each edge in G need have capacity at most k. Hence G' has at most $n^2 k$ edges. Finally, constructing the graph G' and converting the flow to an unsplittable one can be implemented in polynomial time. ◀

As noted previously, Corollary 3.3 above can be used to obtain an edge-confluent solution $(\mathbf{x}', \mathbf{f}')$ which is within a logarithmic factor of $\phi(J, \mathbf{c})$ where $\mathbf{f}$ is a flow in the graph G'. Let $\partial^-(t)$ denote the set of edges going into the sink in the graph G'. For each such unit-capacity edge $e = \{v, t\} \in \partial^-(t)$, look at the flow over this edge according to $(\mathbf{x}', \mathbf{f}')$. Define E_e as the edges this flow uses on its way from the sources to the sink. By the edge-confluence, if $e \neq e'$ are two edges into the sink, then $E_e \cap E_{e'} = \emptyset$. Moreover, since the flow from each source is routed unsplittably, each job j with $x'_j \neq 0$ has all its flow along edges belonging to a unique E_e. Let J_e be the jobs which are routed along $e \in \partial^-(t)$.

We will now present a strategy for the original SSSR instance on graph G which has expected value at least a constant fraction of $\langle \mathbf{w}, \mathbf{x}' \rangle$. This, along with Theorem 2.1 and Corollary 3.3 will imply that its expected value is within a logarithmic factor of the optimal adaptive strategy for the SSSR instance and prove Theorem 1.1. Note that since the sets $(J_e)_{e \in \partial^-(t)}$ form a partition of $\{j \in J \mid x'_j > 0\}$ we have that

$$\langle \mathbf{w}, \mathbf{x}' \rangle = \sum_{j \in J : x'_j > 0} w_j x'_j = \sum_{e \in \partial^-(t)} \sum_{j \in J_e} w_j x'_j$$

Consider some edge e in $\partial^-(t)$. All flow from sources in J_e flows through e, and hence it is the most congested edge among the ones used by sources in J_e. Indeed, $(\mathbf{x}', \mathbf{f}')$ restricted to these sources gives us a solution to (LP_{UFP}) for a single edge – i.e., for the stochastic knapsack instance $\mathcal{I}_K$ with a unit-capacity knapsack, and a set of jobs J_e.

$$\phi_K(J_e, 1) = \max\left\{ \sum_{j \in J_e} w_j x_j \mid \sum_{j \in J_e} x_j \leq 1, \quad x_j \leq \mu_j \; \forall j \in J_e \right\}.$$

Having identified these stochastic knapsack instances, let us state the $O(1)$ approximation for the stochastic knapsack provided by Dean, Goemans and Vondrak [15] which we will use to complete the proof.

▶ **Theorem 3.4** (Stochastic Knapsack [15]). *Given an instance of stochastic knapsack with a set of jobs J' there is a non-adaptive strategy $\mathcal{A}_{DGV}$ which gets expected value at least $\frac{7}{16}\,\phi(J', 1) \geq \frac{7}{32}\,OPT$.*

We now run the non-adaptive algorithm in Theorem 3.4 which attains a value $\Omega(1) \cdot \phi_K(J_e, 1)$. For each source $s_i \in J_e$ routed by this algorithm, we route it along the path from s_i to t taken by the confluent flow $\mathbf{f}'$. Once the cumulative size of the routed jobs exceeds 1, we stop routing jobs from J_e. The NBA implies that the jobs in J_e use up a capacity of at most 2.

Note that interference can occur only between instances corresponding to multiple edges $e' \in G'$ which correspond to the same edge e in the original graph G. However, there are only $K_e = \lceil \lfloor c_e \rfloor / 2 \rceil$ such instances and each instance consumes at most 2 units of capacity, the total capacity used is at most $2K_e \leq \lfloor c_e \rfloor + 1$, and the set of jobs routable in G' are also routable in G. To see this, note that there is 2 units of space for all but the last of the K_e instances, and for the last instance we still have 1 unit of space which is enough to get full value from each job routed successfully by the stochastic knapsack algorithm in Theorem 3.4. Note that the per-instance preemption of jobs is the reason why our strategy is adaptive, and the availability of less than 2 units of space on the last of the K_e instances is the reason that it is unsafe. This completes the proof of Theorem 1.1.

To obtain safe policies for the SSSR under the stronger assumption of the α-NBA, we first use the approach of Theorem 5.2 to get a safe $O(\frac{1}{1-\alpha})$-approximation for stochastic knapsack under the α-NBA. Now using this for each of the stochastic knapsack instances above gives us a safe approximation for SSSR under the α-NBA. Note that under the stronger α-NBA we can afford to choose $K_e = \lfloor c_e \rfloor$ since we use the safe version of the underlying stochastic knapsack algorithm.

To conclude the section on the SSSR, we note that if we could improve the logarithmic factor in Corollary 3.2 to a constant, then we could use our techniques to get a $O(1)$-approximation for the SSSR problem. This would be implied by a stronger conjecture from Chen et al. [12] that says that given any flow in a network with node congestion 1, one can color the sources using a constant number of colors, such that each chromatic class is node-confluently routable with congestion 1.

4 sUFP on Directed Acyclic Graphs

In this section we give an approximation algorithm for the sUFP on DAGs that extends the work of Chekuri et al. [10, Corollary 1.2] which showed an $O(\sqrt{n})$ approximation for the UFP.

The first idea, as with many UFP results, is to divide the jobs into *small* jobs and *large* jobs. Let us define $\delta = 1/8$. Jobs which have an expected size larger than δ are considered large jobs and the rest are small. Let J_s be the set of small jobs and J_ℓ be the set of large jobs. Observe that $\phi(J_s, \mathbf{c}) + \phi(J_\ell, \mathbf{c}) \geq \phi(J, \mathbf{c})$. Now if we could give, for instances consisting exclusively of small and large jobs, non-adaptive algorithms that obtain at least an $1/\gamma_s$-fraction and $1/\gamma_\ell$-fraction of the respective LP values, then choosing the one with higher guaranteed expected value would give us a non-adaptive strategy obtaining expected value at least an $1/(\gamma_s + \gamma_\ell)$-fraction of the optimal adaptive strategy. (See Fact 1.1.)

For large jobs, we will use the existing result by Chekuri et al. [10, Theorem 1.1]. For small jobs, we use the idea of v-separation from this work [10, Section 3.2] together with our confluence-based techniques to obtain a $O(\sqrt{n \log k})$ approximation for the sUFP on DAGs (Theorem 1.2). Recall that n is the number of vertices and k is the number of jobs.

4.1 Routing Large Jobs

Let $\mathcal{I} = (G, \mathbf{c}, J)$ be an instance having optimal payoff OPT where all jobs j satisfy $\mu_j \geq \delta$. For these, define the following instance of the UFP on DAGs: let the edge-capacities become $\widehat{c}_e := \lfloor c_e \rfloor$, and we want to find for each job j a unit-sized s_j-t_j path P_j subject to these capacities to maximize the value of the routed paths. The natural LP relaxation for this problem is:

$$\widehat{\phi}(J, \mathbf{c}) := \qquad \max\left\{ \sum_j v_j x_j \mid \sum_{j: e \in P_j} x_j \leq \widehat{c}_e \quad \forall e \in E, \quad \mathbf{x} \in [0,1]^k \right\}. \qquad (LP_{EDP})$$

The theorem in the work by Chekuri et al. [10, Theorem 1.1] implies that we can find, in polynomial-time, a subset $S \subseteq J$ which is feasible for (LP_{EDP}), such that $\sum_{j \in S} v_j \geq \frac{1}{O(\sqrt{n})} \cdot \widehat{\phi}(J, \widehat{\mathbf{c}})$. For the large jobs, assume that all jobs are unit-sized, find the set S, and try to route each of the jobs in S. The NBA implies that the sizes of the jobs are at most 1, and even unit-sized jobs would not violate the edge-capacities. Hence with probability 1 we get a feasible solution to the stochastic UFP on DAGs with value $\frac{1}{O(\sqrt{n})} \widehat{\phi}(J, \widehat{\mathbf{c}}) \geq \frac{\delta}{O(\sqrt{n})} OPT$. Having shown the approximation result for large jobs, observe that the arguments used above are quite general, and let us record the following theorem for future use.

▶ **Theorem 4.1** (Large Jobs Theorem). *Consider an instance $\mathcal{I} = (G, \mathbf{c}, J_\ell)$ of the* sUFP *(under the* NBA*). Suppose all jobs are δ-large – i.e., they have expected sizes at least $\delta = \delta c_{\min}$. If the integrality gap of the capacitated* EDP *on the graph G is at most γ, then there is a safe*[1] *non-adaptive strategy $\mathcal{A}_\ell$ for* sUFP *that, with probability* 1*, guarantees that value$(\mathcal{A}_\ell) \geq \frac{1}{\gamma} \cdot \widehat{\phi}(J_\ell, \widehat{\mathbf{c}}) \geq \frac{\delta}{4\gamma} \cdot OPT_\ell$.*

4.2 Routing Small Jobs

Let us now examine the case where all jobs are small. The quantity $\phi(J, \mathbf{c})$ represents the weighted flow from the set of sources to the set of sinks. Recall that this quantity is at least half of OPT by Theorem 2.1. We consider all flow paths and partition them into short paths and long paths. For our purposes, paths of length at most $\sqrt{n(1 + \ln k)}$ will be called short paths and the rest will be called long paths. Either the amount of weighted flow along short paths is at least $\phi(J, \mathbf{c})/2$ or that along long paths is at least $\phi(J, \mathbf{c})/2$. We will handle both these cases separately. We will use randomized rounding in both cases.

[1] Note that the strategy described above guarantees that no edge-capacity is violated and is hence safe.

4.2.1 Randomized Rounding for Short Flow Paths

Let $\mathbf{x}$ be the part of the solution to LP_{UFP} which corresponds to the flow along the short paths. If w_i denotes v_i/μ_i then we know that $\sum_{j\in J} w_j x_j$ is at least $\phi(J,\mathbf{c})/2$. We decide to route job j with probability $\alpha x_j/\mu_j$ and job j, if so chosen, is routed on path $P \in \mathcal{P}(s_j,t_j)$ with probability f_P/x_j. With the remaining probability $1-\alpha x_j/\mu_j$ we choose not to route job j. Let Y_{Pj} be the indicator r.v. for whether path P was picked for job j.

We define a random indicator variable Z_{Pj} which indicates if the algorithm $\mathcal{A}$ decides to route job j along path P and there is at least $1/2$ residual capacity on each edge of P at the time of routing it.

$$Z_{Pj} = \begin{cases} 1 & \text{if } Y_{Pj}=1 \text{ and } \sum\limits_{i<j} \sum\limits_{\substack{P'\ni e\\ P'\in\mathcal{P}(s_i,t_i)}} S_i Y_{P'i} \le c_e - 1/2 \text{ for all } e\in P \\ 0 & \text{otherwise} \end{cases}.$$

If we choose $\alpha = \frac{1}{4\sqrt{n(1+\ln k)}}$, we get:

$$\Pr[Z_{Pj}=0 \mid Y_{Pj}=1] \le \sum_{e\in P} \Pr\left[\sum_{i<j} \sum_{\substack{P'\ni e\\ P'\in\mathcal{P}(s_i,t_i)}} S_i Y_{P'i} > c_e/2\right].$$

Note that LP_{UFP} implies

$$\mathbf{E}\left[\sum_{i<j} \sum_{\substack{P'\ni e\\ P'\in\mathcal{P}(s_i,t_i)}} S_i Y_{P'i}\right] = \sum_{i<j} \sum_{\substack{P'\ni e\\ P'\in\mathcal{P}(s_i,t_i)}} \mu_i\cdot(\alpha f_{P'}\ /\mu_i) \le \alpha c_e = \frac{c_e}{4\sqrt{n(1+\ln k)}}.$$

It follows from Markov's inequality that

$$\Pr[Z_{Pj}=0 \mid Y_{Pj}=1] \le \sum_{e\in P} \frac{1}{2\sqrt{n(1+\ln k)}} \le 1/2.$$

We can now complete the proof of the claim, using Markov's inequality once again:

$$\begin{aligned}
&\Pr[\text{value obtained from job } j] \\
&\qquad = v_j \cdot \Pr[\text{Job } j \text{ is successfully routed}] \\
&\qquad = v_j \cdot \sum_P \Big(\Pr[Y_{Pj}=1]\cdot \Pr[Z_{Pj}=1 \mid Y_{Pj}=1] \\
&\qquad\qquad \cdot \Pr[\text{Job } j \text{ is successfully routed along } P \mid Z_{Pj}=1, Y_{Pj}=1]\Big) \\
&\qquad \ge v_j \cdot \sum_P \left(\alpha\cdot\frac{f_P}{\mu_j}\cdot\frac{1}{2}\cdot\frac{3}{4}\right) = \frac{3v_j}{8}\cdot\frac{x_j}{\mu_j}\cdot\frac{1}{4\sqrt{n(1+\ln k)}} = \frac{3w_j x_j}{32\sqrt{n(1+\ln k)}}.
\end{aligned}$$

4.2.2 Randomized Rounding for Long Flow Paths

Recall that long paths are path of length more than $\sqrt{n(1+\ln k)}$. We examine the case where flow of weight at least $\phi(J,\mathbf{c})/2$ is routed along long paths. As in the proof of SSSR in Section 3, we will convert the flow on the given graph to a flow on a corresponding multigraph with edges of unit capacity. As before, we assume without loss of generality (because of the NBA) that the sources s_j are unique vertices, each with a single out-edge of capacity 1 and similarly, that the targets t_j are unique vertices, each with single in-edge of capacity 1.

As before, we define corresponding to each edge e, an integer $K_e = \lceil \lfloor c_e \rfloor / 2 \rceil$. For the given directed acyclic graph $G = (V, E)$, define a (multi)graph $G' = (V, E')$ where for each edge $e = (u, v) \in E$, we have K_e parallel unit-capacity edges (u, v) in E'. Note that $K_e > c_e/3$ for all e and hence the flow along long paths in G, after being scaled down by a factor of 3 can be converted to a flow in G' preserving flow path lengths. Let this flow, which is of weight at least $\phi(J, \mathbf{c})/6$ correspond to the solution $\mathcal{F} = (\mathbf{x}, \mathbf{f})$ to LP_{UFP} for G'.

All flow paths are long and hence the sum over vertices of the weighted flow routed through each vertex is at least $weight(\mathcal{F}) \cdot \sqrt{n(1 + \ln k)}$. Since the total number of vertices is n, there must exist at least one vertex v through which flow of weight at least $\phi(J, \mathbf{c})/6 \cdot \sqrt{(1 + \ln k)/n}$ is routed. Let $\mathcal{F}_v = (\mathbf{x}_v, \mathbf{f}_v)$ be the solution to LP_{UFP} corresponding to the part of $\mathcal{F}$ routed through v. G' is a directed acyclic multigraph and hence the vertex v splits the flow $\mathcal{F}$ into a single-sink instance G'_{in} and a single-source instance G'_{out}. Let us denote the two corresponding flows by $\mathcal{F}_{v,in}$ and $\mathcal{F}_{v,out}$. We infer[2] from Theorem 3.1 that there exists in G'_{in} an edge-confluent flow $\mathcal{F}'_{v,in}$ of weight $weight(\mathcal{F}_{v,in})/(1 + \ln k)$ and in G'_{out} an edge-confluent flow $\mathcal{F}'_{v,out}$ of weight $weight(\mathcal{F}_{v,out})/(1 + \ln k)$. The flow-per-job and hence the weights of $\mathcal{F}_{v,in}$ and $\mathcal{F}_{v,out}$ are same as the corresponding quantities of $\mathcal{F}_v$. Furthermore these quantities are uniformly scaled down by a factor of $(1 + \ln k)$ in the flows $\mathcal{F}'_{v,in}$ and $\mathcal{F}'_{v,out}$. Hence these two flows can be combined to obtain a flow $\mathcal{F}'_v = (\mathbf{x}'_v, \mathbf{f}'_v)$ of weight $weight(\mathcal{F}_v)/(1 + \ln k) \geq \phi(J, \mathbf{c})/(6\sqrt{n(1 + \ln k)})$ which is confluent in both the partitions G'_{in} and G'_{out}.

We will now devise a routing strategy which has expected value within a constant factor of $weight(\mathcal{F}'_v)$ by randomly rounding this flow. Note that despite the confluence properties of the flow $\mathcal{F}'_v$, we cannot directly reduce this v-separable instance to two instances of the Stochastic Knapsack problem as we did in SSSR, because the routings for both parts must synchronize.

We make an initial decision to route job j with probability $\alpha x'_{vj}/\mu_j$ and job j, if so chosen, is routed on path $P \in \mathcal{P}'(s_j, t_j)$ with probability f'_{vP}/x_j. Here $\mathcal{P}'(s_j, t_j)$ denotes the possible set of paths for job j in graph G'. With the remaining probability $1 - \alpha x'_{vj}/\mu_j$ we choose not to route job j. Let Y_{Pj} be the indicator r.v. for whether path P was picked for job j. Again, we define a random indicator variable Z_{Pj} which indicates if the algorithm $\mathcal{A}$ makes an initial decision to route job j along path P and there is at least $1/2$ residual capacity on each edge of P just before making a decision for job j. If $Z_{Pj} = 0$ even though $Y_{Pj} = 1$ the initial decision is overruled and job is not routed. Otherwise the initial decision holds.

$$Z_{Pj} = \begin{cases} 1 & \text{if } Y_{Pj} = 1 \text{ and } \sum\limits_{i<j} \sum\limits_{\substack{P' \ni e \\ P' \in \mathcal{P}(s_i, t_i)}} S_i Z_{P'i} \leq 1/2 \text{ for all } e \in P \\ 0 & \text{otherwise} \end{cases} .$$

We choose $\alpha = 1/8$. The event $\{Z_{Pj} = 0 \mid Y_{Pj} = 1\}$ occurs if there is at least one edge along P which is congested, i.e., it has residual capacity less than $1/2$ at the time of making a decision for job j. Let e_{in} and e_{out} denote the edges on P which are incoming to and outgoing from v. Since $\mathcal{F}'_v$ is confluent in both G'_{in} and G'_{out}, we infer that if at least one

[2] We scale down by a factor of $(1 + \ln k)$ to ensure unit congestion

edge along P is congested, then either e_{in} or e_{out} must be congested. Hence

$$\Pr[Z_{Pj} = 0 \mid Y_{Pj} = 1] \leq \sum_{e \in \{e_{in}, e_{out}\}} \Pr[\sum_{i<j} \sum_{\substack{P' \ni e \\ P' \in \mathcal{P}(s_i,t_i)}} S_i Y_{P'i} > {}^1\!/\!_2].$$

Note that LP_{UFP} implies

$$\mathbf{E}[\sum_{i<j} \sum_{\substack{P' \ni e \\ P' \in \mathcal{P}(s_i,t_i)}} S_i Y_{P'i}] = \sum_{i<j} \sum_{\substack{P' \ni e \\ P' \in \mathcal{P}'(s_i,t_i)}} \mu_i \cdot (\alpha f'_{vP'} / \mu_i) \leq {}^1\!/\!_8.$$

Markov's inequality implies that $\Pr[Z_{Pj} = 0 \mid Y_{Pj} = 1] \leq {}^1\!/\!_2$. We use Markov's inequality again to infer:

$$\begin{aligned}
&\Pr[\text{value obtained from job } j] \\
&\quad = v_j \cdot \Pr[\text{Job } j \text{ is successfully routed}] \\
&\quad = v_j \cdot \sum_P \Big(\Pr[Y_{Pj} = 1] \cdot \Pr[Z_{Pj} = 1 \mid Y_{Pj} = 1] \\
&\qquad\qquad\qquad \cdot \Pr[\text{Job } j \text{ is successfully routed along } P \mid Z_{Pj} = 1, Y_{Pj} = 1] \Big) \\
&\quad \geq v_j \cdot \sum_P \left(\alpha \cdot \frac{f_P}{\mu_j} \cdot \frac{1}{2} \cdot \frac{3}{4} \right) = \frac{3 w_j x_j}{64}.
\end{aligned}$$

Hence the expected value of this rounding strategy is within a constant factor of $weight(\mathcal{F}'_v)$. Finally note that this routing strategy ensures that each edge in G' reaches capacity at most once and hence the capacity consumed on it is at most 2. The definition of G' is such that all but one of the K_e edges in G' will not obstruct each other's jobs from being routed in G if none of the edges ever reaches more than 2 units of congestion and for the last edge the remaining capacity is at least 1, enough to get value from the successfully routed jobs. This completes the proof of Theorem 1.2.

5 Safe Strategies

In this section we address the issue of routing jobs in a way such that we are guaranteed to never overshoot the capacity of any edge. This concept was first studied by Chawla and Roughgarden [8], who called such strategies "safe" strategies. To get non-trivial safe strategies, one has to make an assumption slightly stronger than the NBA. Indeed, we assume that $D_{\max}$, the supremum of the values that any job can take on with non-zero probability, is α where $\alpha \in (0, 1)$ – i.e., the support of each random variable S_j is now $[0, \alpha]$. We refer to this assumption as the α-NBA. As before we have assumed by scaling that $c_{\min} = 1$. Note that we can now get a better upper bound on OPT than what Theorem 2.1 provides : $OPT \leq \phi(J, \mathbf{c} + \alpha\mathbf{1}) \leq (1 + \alpha)\, \phi(J, \mathbf{c})$.

5.1 The Case $\alpha \leq {}^1\!/\!_2$

In case $\alpha \in (0, 1/2]$, any strategy for sUFP that is good with respect to the LP relaxation (LP_{UFP}) can be easily converted to a safe strategy with a loss of a factor of $(1 - \alpha)$.

▶ **Theorem 5.1.** *For $\alpha \in (0, 1/2]$, let instance $\mathcal{I} = (G, \mathbf{c}, J)$ of the* sUFP *satisfy the α-*NBA*. Hence, the instance $\mathcal{I}' = (G, \mathbf{c}(1 - \alpha), J)$ satisfies the* NBA*. Given a strategy $\mathcal{A}$ that is an γ-approximation for the instance $\mathcal{I}'$ w.r.t. the LP relaxation (LP_{UFP}), we can obtain a strategy that is an $\frac{\gamma(1+\alpha)}{1-\alpha}$-approximation for $\mathcal{I}$.*

Proof. Observe that $\phi(J, \mathbf{c}(1-\alpha)) \geq \frac{1-\alpha}{1+\alpha} \cdot \phi(J, \mathbf{c}(1+\alpha))$. We know that the strategy $\mathcal{A}$ for $\mathcal{I}'$ achieves expected value at least $\frac{1}{\gamma} \cdot \phi(J, \mathbf{c}(1-\alpha)) \geq \frac{1-\alpha}{\gamma(1+\alpha)} \cdot OPT(\mathcal{I})$. We claim that this run will not violate the actual capacities $\mathbf{c}$ of the edges. Indeed, we can assume, w.l.o.g., that $\mathcal{A}$ does not route any jobs that use any edges that are already forbidden. Hence, just before an edge capacity is violated in the run of $\mathcal{A}$ on $\mathcal{I}'$, it was used to at most $(1-\alpha)c_e$, and the α-NBA ensures that the job can take on size at most $\alpha \leq \alpha\, c_e$. So the total used-up capacity on each edge e is at most $c_e(1-\alpha) + \alpha \leq c_e$, completing the proof. Note that if $\alpha > 1/2$ the instance $\mathcal{I}'$ does not satisfy the NBA. ◀

5.2 The Case $\alpha \geq 1/2$

In this case we give a reduction that takes an arbitrary strategy for sUFP on *unit-capacity* networks which is good with respect to the LP solution, and transforms it into a safe strategy. We use this e.g., on our result for single-sink UFP.

▶ **Theorem 5.2.** *Let $\alpha \in (0,1)$ and $\gamma \geq 1$. Consider a flow network G with unit-capacity edges; i.e., $\mathbf{c} = \mathbf{1}$. (We allow parallel arcs.) Suppose we have strategy $\widetilde{\mathcal{A}}$ that for any instance $\widetilde{\mathcal{I}} = (G, \mathbf{1}, \widetilde{J})$ of the sUFP on G satisfying the NBA, achieves expected value at least $1/\gamma \cdot \phi(\widetilde{J}, \mathbf{1})$. Then there exists a safe algorithm $\mathcal{A}$ which for all instances $\mathcal{I} = (G, \mathbf{1}, J)$ of the sUFP satisfying the α-NBA achieves expected value at least $\frac{(1-\alpha)}{6\gamma} \cdot \phi(\widetilde{J}, \mathbf{1}) \geq \frac{(1-\alpha)}{6\gamma(1+\alpha)} \cdot OPT$.*

Proof. First, we can assume that for all jobs $j \in J$, we have $\Pr[S_j = 0] = 0$, by losing a factor of 2 in the approximation. To prove this, first imagine there are no jobs having mean size zero, since these can be routed without using any capacity. Let $\mu_{\min} := \min_{j \in J} \mu_j$ be the least mean job size. We transform the jobs in J to be supported on $[\mu_{\min}, \alpha]$ by defining their new size to be $S'_j := \max\{\mu_{\min}, S_j\}$. This increases the mean μ_j of each job j to $\mu_{j'} \leq \mu_j + \mu_{\min} \leq 2\mu_j$, but still satisfies the α-NBA. Consequently, the value of the LP has decreased by at most 2 and our strict positivity assumption is justified. Any strategy safe for the modified instance is also safe for the original instance. An advantage of having strictly positive job sizes is that we can argue that if the given strategy $\widetilde{\mathcal{A}}$ routes a job on some path P, all edges on the path have strictly positive residual capacity (and are not forbidden, of course). If not, if there were some edge of capacity zero, or a forbidden edge, the routing would necessarily be unsuccessful, and we could drop it without any loss in value.

Now define $\alpha' := 1 - \alpha$ and $\delta := \alpha'/2$. Separate the jobs into δ-large (those with $\mu_j \geq \delta$) and δ-small (the remaining). For the δ-large jobs, apply Theorem 4.1 to obtain a safe $(1+\alpha)\gamma/\delta$-approximation. Observe that to apply Theorem 4.1, we need an algorithm for the capacitated UFP problem – however, our assumed algorithm $\widetilde{\mathcal{A}}$ for sUFP is at least as powerful, and hence suffices. (The approximation factor is better than claimed in Theorem 4.1, since (i) we start with unit edge-capacities, and hence we do not need to round down the capacities (ii) We use the better $(1+\alpha)\phi(J, \mathbf{c})$ upper bound on the OPT)

For the δ-small jobs J_s, let us denote the original small instance by $\mathcal{I} = (G, \mathbf{1}, J_s)$, and define a modified instance $\mathcal{I}' = (G, \alpha'\mathbf{1}, J'_s)$. For each job $j \in J_s$, find a threshold $\ell_j \leq \mu_j$ such that

$$\mathbf{E}[\max(\ell_j, \min(S_j, \alpha'))] = \mu_j.$$

In words, we "clip" the job size S_j at α' on the upper side, and at ℓ_j on the lower side, and want the mean to remain unchanged. This is possible since $\mu_j \leq \delta < \alpha'$, so the upper clipping brings the mean down, which the lower clipping can remedy. Now define the size of the new job j' to be $S_{j'} := \max(\ell_j, \min(S_j, \alpha'))$, this clipped random variable, and let $\mu_{j'} := \mathbf{E}[S_{j'}]$.

Observe that $S_j, S_{j'}$ are coupled by definition, such that conditioned on $S_{j'} < \alpha'$, we know that $S_{j'} \geq S_j$.

Observe that the value of the LP relaxation $\phi(J'_s, \alpha'\mathbf{1}) \geq \alpha' \cdot \phi(J_s, \mathbf{1})$. Moreover, the instance $\mathcal{I}' = (G, \alpha'\mathbf{1}, J'_s)$ satisfies the NBA, and hence running $\widetilde{\mathcal{A}}$ on J'_s achieves an expected value of at least $1/\gamma \cdot \phi(J'_s, \alpha'\mathbf{1})$. So we can imagine executing the algorithm $\widetilde{\mathcal{A}}$ on $\mathcal{I}'$, whilst actually routing the jobs in $\mathcal{I}$. I.e., when $\widetilde{\mathcal{A}}$ asks to route job j', we actually run job j, it takes on size S_j, and we report back the size $S_{j'}$ to the algorithm $\widetilde{\mathcal{A}}$. As argued above, we assume that $\widetilde{\mathcal{A}}$ does not route any job on forbidden edges, or edges of zero capacity.

The crucial observation is that conditioned on an edge e's capacity having been used up to less than α', the actual usage (according to the real job sizes S_j) is no more than the usage according to the job sizes $S_{j'}$ reported to $\widetilde{\mathcal{A}}$. This is because, conditioned on $S_{j'} < \alpha'$, we know that $S_{j'} \geq S_j$.

Now to see that this strategy is safe for $\mathcal{I}$, consider an edge on some path P on which $\widetilde{\mathcal{A}}$ routes some job j. Previously e's capacity was used up to less than α' (since it had non-zero residual capacity by our assumption on $\widetilde{\mathcal{A}}$), and even if the current job uses it to its maximum size $\alpha = 1 - \alpha'$, we will not violate the actual capacity. Hence, any job that is routed in $\widetilde{\mathcal{A}}$'s run on $\mathcal{I}'$ will also be successful routed in the run on $\mathcal{I}$.

This gives us an $(1+\alpha)\gamma/\alpha'$-approximation algorithm for small jobs. Using Fact 1.1 the better of the two gives us a $\frac{3\gamma(1+\alpha)}{(1-\alpha)}$-approximation. Moreover, losing another factor of 2 for the transformation to strictly positive job sizes gives us the result. ◀

Theorem 5.2 is the only result in our paper where we require knowing more information about the distribution of S_j beyond just the expectation μ_j. Now we can use Theorems 5.1 and 5.2 to give a safe strategy for variants of the sUFP. In particular, applying this to the stochastic knapsack result from Theorem 3.4 gives us a safe algorithm for that problem, and hence for the SSSR.

6 Conclusions and Discussion

In this paper we gave approximation algorithms for stochastic routing problems under the no-bottleneck assumption. These problems generalize the classical unsplittable flow problem. Our results include improved results for the single-sink case, constant-factor approximations for stochastic routing on trees and paths, and results for general graphs as well. We also gave techniques to convert unsafe strategies into safe ones, for unit capacity networks. Many interesting open questions remain: E.g., can we get a constant-factor for the single-sink setting? Can we give results without the no-bottleneck assumption?

References

1 Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. A mazing 2+ϵ approximation for unsplittable flow on a path. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 26–41, 2014. doi:10.1137/1.9781611973402.3.

2 Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012.

3 Anand Bhalgat. A $(2 + \epsilon)$-approximation algorithm for the stochastic knapsack problem. At `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.7341&rep=rep1&type=pdf`, 2011.

4 Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. Improved approximation results for stochastic knapsack problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1647–1665, 2011.

5 Paul S. Bonsma, Jens Schulz, and Andreas Wiese. A constant factor approximation algorithm for unsplittable flow on paths. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 47–56, 2011. doi:10.1109/FOCS.2011.10.

6 Gruia Calinescu, Amit Chakrabarti, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for resource allocation. *ACM Trans. Algorithms*, 7(4):Art. 48, 7, 2011. doi:10.1145/2000807.2000816.

7 Amit Chakrabarti, Chandra Chekuri, Anupam Gupta, and Amit Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47(1):53–78, 2007. doi:10.1007/s00453-006-1210-5.

8 Shuchi Chawla and Tim Roughgarden. Single-source stochastic routing. In *Proceedings of APPROX*, pages 82–94. Springer, 2006.

9 Chandra Chekuri, Alina Ene, and Nitish Korula. Unsplittable flow in paths and trees and column-restricted packing integer programs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, pages 42–55, 2009. doi:10.1007/978-3-642-03685-9_4.

10 Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. An o(sqrt(n)) approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing*, 2(7):137–146, 2006. doi:10.4086/toc.2006.v002a007.

11 Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Algorithms*, 3(3):Art. 27, 23, 2007. doi:10.1145/1273340.1273343.

12 Jiangzhuo Chen, Robert D. Kleinberg, László Lovász, Rajmohan Rajaraman, Ravi Sundaram, and Adrian Vetta. (Almost) tight bounds and existence theorems for single-commodity confluent flows. *J. ACM*, 54(4):Art. 16, 32 pp. (electronic), 2007. doi:10.1145/1255443.1255444.

13 Jiangzhuo Chen, Rajmohan Rajaraman, and Ravi Sundaram. Meet and merge: approximation algorithms for confluent flows. *J. Comput. System Sci.*, 72(3):468–489, 2006. doi:10.1016/j.jcss.2005.09.009.

14 Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Adaptivity and approximation for stochastic packing problems. In *SODA*, pages 395–404, 2005.

15 Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.*, 33(4):945–964, 2008. doi:10.1287/moor.1080.0330.

16 Yefim Dinitz, Naveen Garg, and Michel X. Goemans. On the single-source unsplittable flow problem. *Combinatorica*, 19(1):17–41, 1999. doi:10.1007/s004930050043.

17 Sudipto Guha and Kamesh Munagala. Approximation algorithms for budgeted learning problems. In *ACM Symposium on Theory of Computing (STOC)*, pages 104–113. ACM, 2007. Full version as *Sequential Design of Experiments via Linear Programming*, http://arxiv.org/abs/0805.2630v1.

18 Sudipto Guha and Kamesh Munagala. Approximation algorithms for bayesian multi-armed bandit problems. *CoRR*, abs/1306.3525, 2013. URL: http://arxiv.org/abs/1306.3525.

19 Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and R. Ravi. Approximation algorithms for correlated knapsacks and non-martingale bandits. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 827–836, 2011.

20 Archit Karandikar. Approximation algorithms for stochastic unsplittable flow problems. Master's thesis, Carnegie Mellon University, 2015. URL: `https://github.com/architkarandikar/MastersThesis`.

21 Jian Li and Wen Yuan. Stochastic combinatorial optimization via poisson approximation. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 971–980, 2013. `doi:10.1145/2488608.2488731`.

22 Will Ma. Improvements and generalizations of stochastic knapsack and multi-armed bandit approximation algorithms: Extended abstract. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1154–1163, 2014. `doi:10.1137/1.9781611973402.85`.

23 F. Bruce Shepherd and Adrian Vetta. The inapproximability of maximum single-sink unsplittable, priority and confluent flow problems. *CoRR*, abs/1504.00627, 2015. URL: `http://arxiv.org/abs/1504.00627`.

24 F. Bruce Shepherd, Adrian Vetta, and Gordon T. Wilfong. Polylogarithmic approximations for the capacitated single-sink confluent flow problem. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 748–758, 2015. `doi:10.1109/FOCS.2015.51`.

A Missing Proofs

A.1 Combining Results for Small and Large Jobs

▶ **Fact 1.1** (Combination). *Consider an instance $\mathcal{I} = (G, \mathbf{c}, J)$ of the* sUFP *with optimal payoff OPT specified by the graph G, edge-capacity vector $\mathbf{c}$ and the set of jobs J. Let J_1 and J_2 form a partition of J and consider instances $\mathcal{I}_1 = (G, \mathbf{c}, J_1)$ and $\mathcal{I}_2 = (G, \mathbf{c}, J_2)$ with optimal payoffs OPT_1 and OPT_2. Suppose for each instance $\mathcal{I}_i$, there exists a polytime non-adaptive algorithm $\mathcal{A}_i$, a polytime computable quantity ξ_i and a quantity $\gamma_i \geq 1$ such that $\mathbf{E}[payoff(\mathcal{A}_i)] \geq \xi_i \geq \frac{1}{\gamma_i} OPT_i$. Then the algorithm $\mathcal{A}$ that returns the solution for the instance $\mathcal{I}_i$ with the higher ξ_i has*

$$\mathbf{E}[payoff(\mathcal{A})] \geq \frac{1}{\gamma_1 + \gamma_2} OPT\,.$$

Proof. Since J_1 and J_2 form a partition of J, $OPT \leq OPT_1 + OPT_2 \leq \gamma_1\xi_1 + \gamma_2\xi_2$. Hence $\max(\xi_1, \xi_2) \geq \frac{OPT}{\gamma_1+\gamma_2}$.

As an example application, suppose we have two different LP rounding algorithms that on instances J_i produce solutions with values $\xi_i = \frac{1}{\gamma_i} OPT_i$. Then taking the larger one is an $(\gamma_1 + \gamma_2)$-approximation. ◀

A.2 Reducing Edge-Confluence to Node-Confluence

The results of Chen et al. [12], along with most other literature address node-confluence. We show how to get Theorem 3.1 on edge-confluence from these resuts. The existing result of Chen et al. that we use is the analog of Theorem 3.1 for node-confluent flows under node-congestion. The node-congestion of a flow is defined as $\textsf{n-cong}(\mathbf{f}) = \max\{1, \max_{v \in V \setminus \{t\}} f_v\}$ where f_v denotes the flow passing though vertex v.

Proof. Theorem 3.1 Consider any digraph $G = (V, E)$ with sources $S_G \subseteq V$, sink node t, and unit edge-capacities, in which we have a general flow $\mathbf{f}$ respecting edge capacities(i.e. $\textsf{cong}(\mathbf{f}) = 1$) that we wish to convert to an edge-confluent flow. We assume w.l.o.g. that this flow is acyclic and that each source has exactly one outgoing edge from it and no edges incoming

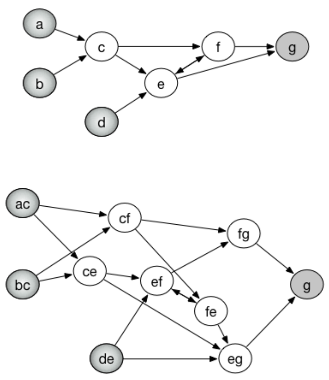

Figure 1 Reducing edge-confluence to node-confluence. Edge-confluent flows in the network above correspond to node-confluent flows in the one below. The sink is shaded grey. The sources are shaded with a gradient.

into it. To justify the latter, note we can augment the graph by adding a new source for each orginal source, connected to it by a unit-capacity edge. Under the NBA, this transformation does not change congestion and flows which are edge-confluent in the augmented graph are also edge-confluent in the original graph.

We construct another graph H with unit node-capacities, which is essentially the directed line graph of G (plus one extra node). This construction is demonstrated in Figure 1. The graph H has a node v_{pq} for every arc (p, q) in G. There is an arc from v_{pq} to v_{rs} exactly when $q = r$. Moreover, it has node v_t, with arcs from all nodes v_{pt} to v_t. Finally, for every source s there is exactly one graph (s, x_s) leaving s as per our assumption. The set of sources in the new graph H is defined by $S_H = \{v_{sx_s} \mid s \in S\}$.

Now given the flow $\mathbf{f}$ in G from sources S to sink t, take any path decomposition of the flow $\mathbf{f}$. Each flow path P can be mapped in a natural way to a flow-path in H: if $P = \langle s, a, b, \ldots, z, t\rangle$, then it is mapped to path $\langle v_{sa}, v_{ab}, \ldots, v_{zt}, v_t\rangle$ in H. Doing this for all flow-paths gives a flow $\mathbf{h}$ in H. The node-capacities in H are satisfied by $\mathbf{h}$ because the edge-capacities in G were satisfied by $\mathbf{f}$. This is a injection from unit edge-congestion flows in G into unit node-congestion flows in H. Note that the procedure can be reversed to obtain a surjection from unit node-congestion flows in H onto unit edge-congestion flows in G. These mappings are not inverses since several flows in H may correspond to a single flow in G[3]. Under these mappings, any edge-confluent flow $\mathbf{f}$ from S_G to t in G is mapped to a node-confluent flow $\mathbf{h}$ from S_H to v_t in H, and vice versa. Since the flow though an edge in G equals flow through the corresponding vertex in H we note that $\mathsf{cong}(\mathbf{f}) = \mathsf{n\text{-}cong}(\mathbf{h})$.

Hence, to prove Theorem 3.1, we do the following: we take the unit-congestion flow $\mathbf{f}$ in G and convert it into a unit-node-congestion flow $\mathbf{h}$ in H. Now we can use results of Chen et al. [12] on node-confluent flows in unit-capacity graphs. They show how to convert $\mathbf{h}$ into:

- A node-confluent flow $\mathbf{h}'$ with $\mathsf{n\text{-}cong}(\mathbf{h}') = 1 + \ln k$.
- A node-confluent flow $\mathbf{h}''$ respecting node-capacities (i.e. $\mathsf{n\text{-}cong}(\mathbf{h}'') = 1$) that routes flow from a subset of the sources in S_H having total flow at least a third of the total original flow in $\mathbf{h}$.

Mapping these flows back to G gives us the flows claimed in Theorem 3.1. ◀

[3] Consider node c in Figure 1 and note that $(a \to c \to e, b \to c \to f)$ and $(a \to c \to f, b \to c \to e)$ can be two different alternatives for path decomposition.

A reduction from node-confluence to edge-confluence is easy as Chen et al. [13] had previously observed. This result thus identifies an interconvertability between node-confluence and edge-confluence. Note that we have used this construction for unit capacity networks only since it suffices our purpose of addressing the SSSR. However this argument also extends in a straightforward way to capacitated graphs. Hence it can also be used to obtain a $O(\log^6 n)$ approximate edge-confluent flow of congestion 2 corresponding to the recent node-confluence results of Shepherd, Vetta, and Wilfong [24, Theorem I.7].