MATHEMATICS OF OPERATIONS RESEARCH
Vol. 18, No. 1, February 1993
Printed in U.S.A.

PROBABILISTIC ANALYSIS OF NETWORK FLOW ALGORITHMS

RICHARD M. KARP, RAJEEV MOTWANI AND NOAM NISAN

This paper is concerned with the design and probabilistic analysis of algorithms for the maximum-flow problem and capacitated transportation problems. These algorithms run in linear time and, under certain assumptions about the probability distribution of edge capacities, obtain an optimal solution with high probability. The design of our algorithms is based on the following general method, which we call the mimicking method, for solving problems in which some of the input data are deterministic and some are random with a known distribution:

1. Replace each random variable in the problem by its expectation; this gives a deterministic problem instance that has a special form, making it particularly easy to solve;
2. Solve the resulting deterministic problem instance;
3. Taking into account the actual values of the random variables, mimic the solution of the deterministic instance to obtain a near-optimal solution to the original problem;
4. Fine-tune this suboptimal solution to obtain an optimal solution.

We present linear time algorithms to compute a feasible flow in directed and undirected capacitated transportation problem instances. The algorithms are shown to be successful with high probability when the probability distribution of the input data satisfies certain assumptions. We also consider the maximum flow problem with multiple sources and sinks. We show that with high probability the minimum cut isolates either the sources or the sinks, and we give a linear-time algorithm that produces a maximum flow with high probability.

1. Introduction. Probabilistic analysis of combinatorial problems has been the subject of many recent investigations [KLMR]. The starting point for such analysis is the assumption that the problem instances are drawn from a probability distribution. Under this assumption one studies the behavior of the solution to the combinatorial problem or the performance of some algorithm. The analysis often establishes that certain quick-but-dirty algorithms produce optimal or near-optimal solutions with high probability. In this paper we will be concerned with devising quick-but-dirty algorithms of this type for some combinatorial problems.

The problems considered here are all closely related to the maximum flow problem. In particular, we present fast (linear time) algorithms for the maximum flow problem and certain versions of the transportation problem; these algorithms are guaranteed to succeed with high probability if the probability distribution of the inputs satisfies certain assumptions. A key part of our work is the formulation and application of a new technique for solving problems with probabilistic inputs. We call this technique the probabilistic mimicking of deterministic solutions. The mimicking paradigm works as follows:

Stage 1 (Deterministic Relaxation). Suppose we are given a problem instance $P(\vec{X},\vec{Y})$ with parameters (or input) $\vec{X}$ and $\vec{Y}$. The vector $\vec{X}$ represents the random values and the vector $\vec{Y}$ represents the deterministic values. Construct a deterministic

Received January 11, 1988; revised June 28, 1991.
AMS 1980 subject classification. Primary: 90B15. Secondary: 90C35.
IAOR 1973 subject classification. Main: Networks. Cross references: Programming, Transportation.
OR/MS Index 1978 subject classification. Primary: 484 Networks/Flow Algorithms. Secondary: 488 Networks/Stochastic.
Key words. Probabilistic analysis of algorithms, maximum flow problem, transportation problem.

0364-765X/93/1801/0071/$01.25

relaxation of P by replacing each random variable by its *expected* value. We now have a deterministic problem instance $P(\mathrm{Exp}[\vec{X}], \vec{Y})$.

Stage 2 (Solution of Deterministic Relaxation). Suppose we are given a problem instance $P(\mathrm{Exp}[\vec{X}], \vec{Y})$. This instance is typically of a special form for which a highly efficient algorithm is available.

Stage 3 (Mimicking Process). Construct a solution to the original problem instance $P(\vec{X}, \vec{Y})$ by *mimicking* the solution constructed in the previous step. The exact form of the mimicking process will depend on the problem under consideration. The idea is to use the solution of the deterministic relaxation as a guide in solving the original problem instance.

Stage 4 (Fine Tuning). In this stage we fine-tune the solution obtained in the previous stage to come up with an optimal solution to the original problem instance. The fine-tuning process is highly efficient because, with high probability, the solution resulting from the mimicking process is already close to the optimal.

The paper is organized as follows. The rest of this section is devoted to the definition of the problems under consideration and a brief discussion of our results, as well as their relation to the previous work in this area. In §2 we present some preliminary results which will prove useful in deriving our main theorems. In particular, we will describe a fine-tuning algorithm which will implement the last stage of our paradigm. In §3 we present algorithms for the undirected transportation problem and the max-flow problem. In §4 we present an algorithm for the directed transportation problem. Finally, in §5, we discuss further work along these lines.

1.1. *Problem definitions.* We start by defining the three problems under consideration. These are the Maximum Flow Problem, the Supply Demand Problem and the Capacitated Transportation Problem. We also present the classic minmax theorems associated with each of these problems (see [La] for more details). The problems are defined for the case of directed graphs (digraphs) only. These definitions apply to undirected cases also if we look upon each undirected edge (u, v) as representing the two directed edges (u, v) and (v, u).

Maximum Flow Problem. Let G be a digraph with vertices $V = S \cup I \cup T$, where S represents the *source* vertices, T represents the *sink* vertices and I represents the *intermediate* vertices. Let E represent the set of directed arcs in the digraph G. A *capacity* function, c: $E \to \Re^+$, assigns a nonnegative real number $c(u, v)$ to every arc (u, v) in E. An instance of the max-flow problem consists of a digraph D and a capacity function c.

Given an instance of a max-flow problem a function, f: $E \to \Re^+$ is called a *flow* function if it satisfies the following constraints:

$$0 \le f(u,v) \le c(u,v), \quad \forall (u,v) \in E,$$

$$\sum_{(v,w)\in E} f(v,w) = \sum_{(w,v)\in E} f(w,v), \quad \forall v \in I.$$

Let $f(A, B) = \sum_{a \in A,\ b \in B} f(a, b)$ where $A, B \subset V$. The value of a flow function, $\mathrm{val}(f)$, can be defined as follows:

$$\mathrm{val}(f) = f(S, V-S) - f(V-S, S) = f(V-T, T) - f(T, V-T).$$

The max-flow problem is to find a maximum-value flow function on a given instance of a flow problem.

This problem has been studied quite extensively and the fastest known algorithm is due to Alon [A1] (see also [GT, CH]) which runs in $O(nm \log n)$ time, where $n = |V|$ and $m = |E|$. In the case of dense graphs the best max-flow algorithm known requires $O(n^3)$ time. We assume that the reader is familiar with the basic theory of network flows. This includes the notion of a *cut* and the *capacity* of a cut, as well as the following theorem.

THEOREM 1.1 (Max-Flow Min-Cut Theorem). *The maximum value of an (S, T)-flow is equal to the minimum capacity of an (S, T)-cut.*

Supply-Demand Problem. An instance of the supply-demand problem consists of a digraph $G(V, E)$ and a capacity function c as before, and also a collection of supplies and demands associated with the sources and the sinks, respectively. Associated with every source vertex $s \in S$ is a nonnegative number a_s called the *supply* at s. Similarly, associated with every *sink* vertex $t \in T$ is a nonnegative number b_t called the *demand* at t. The supply-demand problem is to find a *feasible flow*, viz., a flow which meets the demands at T from the supplies available at S. More formally, we are looking for a flow $f: E \to \Re^+$ which satisfies the following conditions:

$$0 \leqslant f(u,v) \leqslant c(u,v), \quad \forall u,v \in V,$$

$$f(s, V - \{s\}) - f(V - \{s\}, s) \leqslant a_s, \quad \forall s \in S,$$

$$f(v, V - \{v\}) - f(V - \{v\}, v) = 0, \quad \forall v \in I,$$

$$f(V - \{t\}, t) - f(t, V - \{t\}) \geqslant b_t, \quad \forall t \in T.$$

Let $a(X) = \Sigma_{v \in X} a_v$ when $X \subset S$, similarly define $b(X) = \Sigma_{v \in X} b_v$ for $X \subset T$. Let $c(X, \bar{X})$ denote the capacity of the $(X, \bar{X})$ cut in G. The following theorem [*La*] *gives necessary and sufficient conditions for a feasible flow to exist in an instance of the supply-demand problem.*

THEOREM 1.2 (Supply-Demand Theorem). *A feasible flow exists in the supply-demand problem if and only if the following inequality holds for every subset $X \subset V$:*

$$b(T \cap \bar{X}) - a(S \cap \bar{X}) \leqslant c(X, \bar{X}).$$

This theorem requires that for every $X \subset V$ the excess of demands over supplies in $\bar{X}$ must be less than the total capacity of the edges leading out of X. There is an easy transformation from an instance of a supply-demand problem to an instance of a max-flow problem and vice versa.

Capacitated Transportation Problem. The capacitated transportation problem is a special case of the supply-demand problem. In this problem we have no intermediate vertices, i.e., $I = \varnothing$, and the graph is bipartite between S and T. We will only consider the cases where the sum of all the supplies is equal to the sum of all the demands. The Supply-Demand Theorem when applied to the transportation problem yields the following:

THEOREM 1.3 (Transportation Theorem). *A feasible flow exists in an instance of the transportation problem if and only if the following condition holds for every subset $X \subset S$:*

$$\sum_{x \in X} a_x \leqslant \sum_{t \in T} \min(b_t, c(X, \{t\})).$$

Probabilistic formulation. For each of these problems we will make two probabilistic assumptions:

1. each edge is present in the graph with probability $p(n)$, independent of the other edges.

2. the capacities of the edges are i.i.d. random variables which have bounded support.

In certain cases the probabilistic assumptions will only apply to some subset of the edges in the complete graph. For example, we will assume that a certain subset of the edges is always present (or absent) while the rest satisfy the first probabilistic assumption. We will also consider the case where certain edges have fixed (or deterministic) capacities while the rest satisfy the second probabilistic assumption.

1.2. *Previous work and main results.* The study of random graphs was initiated by Erdos and Renyi in 1959 [ER1]. The theory of random graphs is concerned with graphs drawn from certain probability spaces. A well-studied random graph [Bo] is called $G_{n,p}$, where n is a positive integer and $0 \leqslant p(n) \leqslant 1$. The probability space consists of all graphs on the vertex set $V = \{1, 2, \ldots, n\}$. In the graph $G_{n,p}$, the probability that an edge is present is p independently for each edge. In other words, the probability of any graph with e edges is $p^e(1-p)^{N-e}$, where $N = n(n-1)/2$. Similarly, we define the random graph $B_{n,p}$ which is chosen from the space of all bipartite graphs on the vertex set $V = S \cup T$, where $|S| = |T| = n$. Independently for each edge, the probability that the graph $B_{n,p}$ contains that edge is p.

We shall require methods for finding perfect matchings in random graphs. A *perfect matching* in an undirected graph $G = (V, E)$ is a spanning subgraph in which each vertex in V has degree one. Consider an instance of the transportation problem in which the underlying bipartite graph has all its edges directed from the sources to the sinks. The problem of finding a perfect matching in a bipartite graph is the special case of this directed transportation problem where all the supplies, demands and capacities are set to one. The problem of finding a perfect matching in a random bipartite graph $B_{n,p}$ is a special case of the probabilistic transportation problem. Erdös and Rényi [ER2, ER3] proved the following theorem about the existence of perfect matchings in random bipartite graphs.

THEOREM 1.4. *Let $p(n) = (\ln n + c)/(n-1)$, where c is some constant. Then, for $B = B_{n,p}$,*

$$\lim_{n\to\infty} \text{Prob}\,[\textit{B has a perfect matching}\,] = e^{-2e^{-c}}$$

Angluin and Valiant [AV] gave a fast algorithm, called the Proposal Algorithm, to construct a bipartite matching. This algorithm works in $O(n \log n)$ time and, with high probability, constructs a perfect matching in a random bipartite graph, provided that $p(n)$ is large enough. They obtained the following result.

THEOREM 1.5. *For all $\alpha > 0$, there exists $\beta > 0$ such that if $p(n) > \beta \log n/n$ then, for $B = B_{n,p}$,*

$$\text{Prob}\,[\textit{Proposal Algorithm fails on B}\,] = O(n^{-\alpha}).$$

Goldschmidt and Hochbaum [GH] gave a greedy algorithm which improves upon the running time of the Proposal Algorithm. The greedy algorithm works for random graphs where $p > \gamma \ln n/n$ and, with high probability, computes a perfect matching in time $O(n \log(1/p))$. This algorithm uses the Proposal Algorithm as a subroutine and the constant γ is larger than the constant in the result of Angluin and Valiant.

However, unlike the Proposal Algorithm, the greedy algorithm has a running time which decreases with increasing density of the random graph. A different type of result was obtained by Motwani [Mo1, Mo2] who showed that the $O(\sqrt{n}\, m)$ algorithms for bipartite and nonbipartite matchings perform exceedingly well when the input is a random graph. In particular, it was shown that if the input random graph has $p(n) > \log n/n$ then these deterministic matching algorithms terminate in time $O(m \log n/\log(np))$ time. Even though this running time is strictly greater than that of the Proposal Algorithm or the Greedy Algorithm, these results apply to a wider class of random graphs.

Consider now the Max-Flow problem where $|S| = |T| = 1$ and the edge capacities are i.i.d. random variables. This problem was considered earlier by Frank and Hakimi [FrHa] and Frank and Frisch [FrFr]. They studied the random variable $\mathrm{val}(f)$ under the above assumptions and obtained results about its probability distribution. Karp [Ka], Grimmett and Welsh [GW] and Grimmett and Suen [GS] obtained strong asymptotic results for complete graphs with i.i.d. edge capacities. In particular, they showed that the minimum cut is almost surely the set of edges incident on the source or those incident on the sink. These results are all existential and do not yield any fast algorithms to construct the maximum flow.

Doulliez and Jamoulle [DJ] proposed a decomposition method to determine the existence of a feasible flow in an instance of the transportation problem when the edge capacities are i.i.d. random variables. There are also results concerning bounds on the probability of existence of a feasible flow in probabilistic transportation problems [PB]. Hassin and Zemel [HZ] studied the probabilistic version of the transportation problem where the underlying graph is complete and the edge capacities are random variables. A collection of random variables $\{c_k\colon k \in K\}$ is said to be *proper* with respect to a constant α if, for each $y \geq 0$ and each $k \in K$, $\mathrm{Prob}[c_k \geqslant y | S] \leqslant \alpha y$. Here S represents any conditioning event concerning the variables $\{c_l\colon l \in K, l \neq k\}$. Hassin and Zemel considered the case where the edge capacities form a proper collection and the supplies/demands are bounded from above. In effect, the "properness" condition requires that each edge capacity be positive. This means that the underlying graph must be complete. Under these assumptions, they presented an algorithm which, with high probability, computes a feasible flow in linear time. Hochbaum [Ho] studied the 0/1 maximum flow problem under the assumption that the underlying graph is chosen from $G_{n,p}$, where $p(n) = \Omega(\sqrt{n \log n}\,/n)$. Since all edge capacities are either 0 or 1, this problem is equivalent to that of finding a maximum collection of vertex-disjoint paths from source to sinks. Hochbaum presented a sublinear time heuristic algorithm which, with high probability, computes a maximum flow in the random instances described above.

Our main results are as follows (some of these results are from the second author's dissertation [Mo1]). We first consider the undirected transportation problem where the edge capacities are random variables and there is a bound on the size of the supplies/demands. We also consider the directed transportation problem where the edge capacities are random variables and the supplies/demands satisfy a *realizability* condition. For both kinds of transportation problems we present linear time algorithms which compute a feasible flow with high probability. We look at the max-flow problem where the edge capacities are i.i.d. random variables. Here we show that the minimum cut is almost surely the cut isolating the sources or the sinks. Again, we present a linear time algorithm that solves such max-flow problem instances, with high probability. Our results are essentially the best possible under the natural probabilistic assumptions being made. For example, in the case of the transportation problems we assume that the demand at a node is smaller than the expected capacity of the edges coming into it by a small factor (roughly $\log n$). For significantly larger demands it is not possible to guarantee that the problem has a feasible solution with reasonably high probability.

2. Preliminaries.

In this section we describe two special cases of the flow problems under consideration. The first problem is concerned with the supply-demand problem in a nonprobabilistic setting where all capacities are $0/1$. The second problem is a special case which arises in the last stage of our mimicking paradigm, viz., the fine-tuning stage. We present an algorithm for handling this version of the problem.

2.1. *Realization of* $0/1$ *matrices.* The following notion of realizability of $0/1$ matrices will prove useful in the description of our algorithms Let $R = (r_1, r_2, \ldots, r_m)$ and $C = (c_1, c_2, \ldots, c_n)$ denote two vectors with nonnegative integral entries. The pair (R, C) is said to be *realizable* if and only if there exists a $0/1$ $m \times n$ matrix $M = M(R, C)$ with R and C as its row-sum and column-sum vectors, respectively. The matrix $M(R, C)$ is called a *realization* of the pair (R, C). The realizability problem is to compute the realization of a pair of row-sum and column-sum vectors.

The realizability problem is closely related to the capacitated transportation problem. Consider instances of the transportation problem where the underlying bipartite graph is complete with all edges directed from S to T and of capacity one. It is easily seen that finding a feasible flow in such transportation problem instances with the integral supply-demand vectors (A, B) is equivalent to finding the realization of the pair (A, B). Ryser [Ry] and Gale [Ga] gave necessary and sufficient conditions for a pair (R, C) to be realizable. There is a simple greedy algorithm [FoFu, FR, Ga] to construct the realization matrix provided it is feasible. This algorithm works in linear time.

A variant of the realization problem is where all the diagonal entries in the matrix are unbounded (positive integers) and while the off-diagonal entries are required to be $0/1$. Let $M_\infty(R, C)$ denote a realization of the row and column sums as a matrix where the diagonal entries are unbounded. Motwani [Mo3] provided a nearly-linear time algorithm to construct $M_\infty(R, C)$, when feasible.

Another notion of realizability will also be used in our algorithms. Let R and C be as before. Also, let $\overline{R} = R - D.\vec{1}$ and $\overline{C} = C - D.\vec{1}$, where D is a positive integer and and $\vec{1}$ represents the vector with a 1 in each entry. The pair (R, C) is said to be *D-realizable* if the pair $(\overline{R}, \overline{C})$ is *realizable*. Finally, we will call a pair (R, C) as being (D, ∞)-realizable if the pair $(R - D.\vec{1}, C - D.\vec{1})$ has a realization with unbounded diagonals. A *D-realization* of the pair (R, C) will be denoted by $M_D(R, C) = M(\overline{R}, \overline{C})$.

2.2. *A fine-tuning algorithm.* The final stage of our mimicking paradigm involves the fine-tuning of the solution obtained by the mimicking process. The algorithm described below will be used for this purpose. Consider an instance of the transportation problem with the underlying undirected bipartite graph $G(S \cup T, E)$ satisfying the following conditions:

(a) $|S| = |T| = n$.

(b) for each $i \in S$ and $j \in T$, the supply a_i and the demand b_j are positive integers such that $0 < L(n) \leqslant a_i$ and $b_j \leqslant U(n) < \sqrt{(nL(n)/\log n)}$.

(c) $\Sigma_{i \in S} a_i = \Sigma_{j \in T} b_j$.

(d) each edge, independently, is present with probability $p(n)$ and has capacity 1.

Here L denotes some arbitrary function of n. Note the above conditions imply that $0 < L, U \leqslant n/\log n$; for supplies and demands much larger than this it is impossible to guarantee that a feasible solution exists with high probability.

The following algorithm uses the Proposal Algorithm [AV] as a subroutine. It first decomposes the transportation problem instance into $U(n)$ instances of the matching problem for random bipartite graphs. The algorithm constructs $U(n)$ bipartite graphs ensuring that each vertex occurs in exactly as many of these graphs as its supply or demand. The edges from the graph G are equiprobably distributed among the $U(n)$ subgraphs. It is shown that, with high probability, the Proposal Algorithm will find a perfect matching in each of these subgraphs. The algorithm sends one unit of flow along each edge which is in the union of the perfect matchings found by the Proposal Algorithm. This is the required flow.

Let $A_i = \Sigma_{r=1}^{i} a_r$ and $B_j = \Sigma_{r=1}^{j} b_r$, where $A_0 = B_0 = 0$.

The Fine-Tuning Algorithm.

Step (1). Construct U bipartite graphs $G^0, \dots, G^{U-1}$ with vertex sets $S^0, \dots, S^{U-1}$ and $T^0, \dots, T^{U-1}$ as follows. Place each vertex $s_i \in S$ in the vertex sets $S^{(A_{i-1}+1) \bmod U}, \dots, S^{A_i \bmod U}$. Similarly, place each vertex $t_j \in T$ in the vertex sets $T^{(B_{j-1}+1) \bmod U}, \dots, T^{B_j \bmod U}$.

Step (2). Color each edge $(s_i, t_j) \in E$ independently and equiprobably with one of U colors.

Step (3). for each edge $(s_i, t_j) \in E$, place it in $E(G^k)$ if and only if it has color k and $s_i \in S^k$, $t_j \in T^k$.

Step (4). Using the Proposal Algorithm, find a perfect matching in each of the U subgraphs generated in the previous steps. Let $F \subseteq E$ denote the union of the U perfect matchings.

Step (5). Saturate in the forward direction all the edges in F. This flow will be a feasible solution to the transportation problem instance under consideration.

The following theorem results.

THEOREM 2.1 (Fine-Tuning Theorem). *For all $\alpha > 0$, there exists $\beta > 0$ such that $p(n) > \beta U^2 \log n/nL$ implies that the Fine-Tuning Algorithm finds a feasible flow in the transportation problem instance in $O(n^2)$ time with probability $1 - O(U(U/nL)^\alpha)$.*

PROOF. The following claims will constitute the proof of this theorem. First, note that our construction ensures that the number of sources and sinks is equal in each of the U graphs being constructed. Let n_k denote the number of sources or the number of sinks in the graph G^k, i.e., $n_k = |S^k| = |T^k|$.

Claim 2.1. $\lfloor nL/U \rfloor \leqslant n_k \leqslant n$, $\forall k \in [0, U-1]$.

The validity of Claim 2.1 is established as follows. Clearly, the number of source nodes in each graph G^k is at least $\lfloor A_n/U \rfloor$, where A_n is the sum of all the supplies. By condition (b), we have that A_n must be at least nL and this implies the desired result. A similar argument works for the case of the sink vertices.

Claim 2.2. $\lim_{n\to\infty} n_k = \infty$.

Claim 2.2 is proved as follows. By Claim 2.1 and condition (b), we have that n_k is at least $\lfloor nL/U \rfloor$. We are also given that $U < \sqrt{(nL/\log n)}$. We conclude that $n_k > nL/U > U \log n$. Since $U > 0$, the desired result follows. The next claim follows from the construction described in Step (1).

Claim 2.3. *Each $s_i(t_j)$ occurs once each in exactly $a_i(b_j)$ of the U subgraphs constructed in Step (1).*

It is clear that in each of the U subgraphs the edges are present independently of other edges in the same subgraph although there is dependence between two such subgraphs. Let p_k be the probability that an edge is present in the graph G^k. We are given that $p(n) > \beta U^2 \log n/nL$. Since each edge is placed in one of U graphs chosen uniformly at random, we have the following claim.

Claim 2.4.

$$p_k = \frac{p(n)}{U(n)} \geqslant \beta \frac{\log n_k}{n_k}.$$

The Proposal Algorithm will succeed in finding a perfect matching in G^k in $O(n \log n)$ time with probability $1 - O((U/nL)^\alpha)$, when $p_k > \beta \log n_k / n_k$. The relation between α and β is the same as for the Proposal Algorithm. The probability that the Proposal Algorithm does not succeed on all G^k is bounded from above by the sum of the probabilities of failure on each of the U matching problems. This probability can be seen to be suitably bounded.

The running time of the entire process can be determined as follows. Step (1) takes time $O(Un)$ and this is clearly $O(n^2)$. Since the number of edges is at most n^2, Step (2) takes time $O(n^2)$. Each call to the Proposal Algorithm takes time $O(n \log n)$ and there are at most $U(n) \leqslant n/\log n$ such calls. Thus, Step (3) also takes time $O(n^2)$. The time required for Step (4) is proportional to the total supply at the source nodes. This quantity is $O(n^2)$. □

We make two remarks about the generality of this algorithm. First, note that the algorithm uses each edge in the forward direction only. The algorithm would work equally well in the case where the edges are directed, though it would not send any flow along edges which are directed from T to S. Also, the algorithm does not actually require that $|S| = |T| = n$. A careful examination of the proof shows that it would be sufficient to have $U^2 \log n / Lp(n) \leqslant |S|, |T| \leqslant n$, where n is now the size of the larger of the two vertex sets. We need to impose these bounds on the sizes of S and T to ensure that n_k (in Claim 2.1) cannot be too small or too large.

We could have used the Greedy Algorithm due to Goldschmidt and Hochbaum [GH] instead of the Proposal Algorithm to obtain the Fine-Tuning Algorithm. In fact, as the analysis of Motwani [Mo2] shows, we could have used Dinic's algorithm [Di] for bipartite matchings as well. Using the Greedy Algorithm will, in general, improve the running time of the Fine-Tuning Algorithm described above. However, it turns out that in the applications of the Fine-Tuning Algorithm described in later sections the overall running times are not affected by the use of the Greedy Algorithm instead of the Proposal Algorithm. The use of Dinic's algorithm would have allowed us to apply the Fine-Tuning Algorithm to graphs with even smaller density. Again, this is not really required in the algorithms which will be presented below.

3. The undirected transportation problem.

We now present an algorithm to solve certain instances of the transportation problem where the underlying graph is undirected. Let $D(n)$ be a positive function such that $D(n) < n/2\sqrt{\log n}$. Consider an instance of the transportation problem which satisfies the following conditions:

(a) the underlying graph $G(S \cup T, E)$ is undirected and $|S| = |T| = n$.

(b) the probability that an edge from $S \times T$ is present is p (a positive constant), and if an edge is present then its capacity is 1.

(c) $\Sigma_{u \in S} a_u = \Sigma_{v \in T} b_v$, and all supplies and demands are nonnegative integers.

(d) $\forall u \in S$, $a_u < pD(n)$ and $\forall v \in T$, $b_v < pD(n)$.

We will show that the transportation problem instances satisfying these conditions are feasible with high probability. This will be done constructively by specifying a linear time algorithm which succeeds in finding the feasible flow with high probability. Once again note that the supply or demand at a node is only slightly smaller than the expected capacity of the edges incident at it. This is essential to guarantee feasibility. The following theorem results.

THEOREM 3.1 (Undirected Transportation Theorem). *The Undirected Transportation Algorithm finds a feasible flow for transportation problem instances satisfying conditions* (a)–(d) *in linear time with probability* $1 - O(n^{-\gamma})$, *where* $\gamma > 0$ *is a constant which depends on* p.

The expected value of the capacity of any edge is p. Consider the deterministic relaxation of the above problem. It would correspond to finding a feasible flow for the supplies a_i and the demands b_j in the case where the underlying undirected graph is complete with all edges having capacity p. It is not very hard to show that the deterministic problem has a feasible solution. However, we will need to find a feasible flow using the edges in the forward direction only. In this directed case, the problem need not have a feasible solution at all, e.g., consider the case where $a_1 = b_1 = 2$ and all other supplies and demands are 0. To get around this problem we will add a large number $E(n)$ to each supply and demand. Now we will be able to find a feasible flow for the deterministic relaxation using edges in the forward direction only. It has been observed [HZ] that a transportation problem instance is feasible provided the values of the supplies and demands are sufficiently uniform. Uniformity of the supplies and demands requires that the supply (demand) at a source (sink) is in proportion to the net capacity of the edges incident at that source (sink). In a sense, adding $E(n)$ to each supply and demand corresponds to making their values uniform, since the expected total capacity of the edges incident at each vertex is equal.

The Undirected Transportation Algorithm is based on the mimicking paradigm. The first stage of this algorithm constructs a solution to the deterministic relaxation of the original problem. To simplify the algorithm, we will scale up the deterministic relaxation by a factor of $c = 1/p$. This corresponds to multiplying all capacities, supplies and demands by a factor of c. The original problem instance does not need to be scaled. Let us choose $E(n) = n/2\log n$. Define $a'_u = ca_u + E(n)$ for all $u \in S$, and $b'_v = cb_v + E(n)$ for all $v \in T$. Also, let a' be the s-dimensional vector (a'_u) and b' the t-dimensional vector (b'_v). The deterministic problem is an undirected transportation problem with (a', b') as the supply-demand vectors and all edges of capacity $cp = 1$. We will use all edges in the forward direction only and so the deterministic problem is exactly that of finding a realization of (a', b').

The first stage of the algorithm constructs a realization of the pair (a', b'), viz., the $0/1$ $n \times n$ matrix $M(a', b')$. This corresponds to the deterministic solution of the transportation problem instance with all random variables (in this case, the edge capacities) replaced by their expected values. In the second step we mimic the solution of the deterministic relaxation by saturating the edges (if present) which correspond to the nonzero entries in $M(a', b')$. The flow is sent in the forward direction, i.e., from S to T. Finally, using the edges in the backward direction we fine-tune the solution to obtain a feasible flow for the transportation problem instance under consideration. The fine-tuning will take care of the error introduced at the mimicking stage, as well as the any extra flow caused by the addition of $E(n)$ to the supplies and demands in the scaled version of the deterministic relaxation.

The Undirected Transportation Algorithm.

Step (1) [Deterministic Relaxation]. Construct the $0/1$ $n \times n$ matrix $M(a', b')$. The pair (a', b') must have only integral entries for the realization to be feasible. This can be ensured by rounding up or rounding down each a'_u, b'_v while still satisfying the condition $\sum_{u \in S} a'_u = \sum_{v \in T} b'_v$.

Step (2) [Mimicking Process]. Saturate in the forward direction all existing edges which correspond to the 1's in $M(a', b')$. This yields a $0/1$ $n \times n$ flow matrix N. Let the row-sum and the column-sum vectors of this matrix be $\bar{a}$ and $\bar{b}$, respectively.

Step (3) [Fine-Tuning]. At this point each $u \in S$ has sent an excess of $\bar{a}_u - a_u$ units of flow, while each $v \in T$ has received an excess of $\bar{b}_v - b_v$ units of flow. Using the Fine-Tuning algorithm and all edges in the backward direction, send out $\bar{b}_v - b_v$ units of flow from each $v \in T$. Ensure that each $u \in S$ receives exactly $\bar{a}_u - a_u$ units of flow.

Step (4). Combine the flows constructed in the two previous steps to obtain the desired feasible flow.

3.1. *Analysis of the Undirected Transportation Algorithm.* We will prove the Undirected Transportation Theorem with the help of the lemmata presented below. Lemma 3.1 establishes that the first step of the algorithm will succeed by showing that the pair (a', b') is realizable. The realization algorithm will construct the matrix $M(a',b')$ in linear time given its feasibility.

LEMMA 3.1. *The pair (a', b') is realizable.*

PROOF. Consider a transportation problem instance $\mathscr{I}$ with $|S| = |T| = n$. Assume that all edges from $S \times T$ are present. Let each edge have capacity 1 and be directed from S to T. Let a'_u be the supply at $u \in S$, and b'_v the demand at $v \in T$. The Integrality Theorem for flow problems can be applied to this transportation problem instance. It implies that, in a feasible transportation problem instance, if all supplies, demands and capacities are integral then there is an integral feasible flow. The rounding process in Step (1) ensures that the supplies and demands are integral, while all capacities are 1 in $\mathscr{I}$. Thus, we have that $\mathscr{I}$ is feasible if and only if the pair (a', b') is realizable. We invoke the Transportation Theorem to establish the feasibility of $\mathscr{I}$.

By the Transportation Theorem, $\mathscr{I}$ is feasible if and only if $\forall X \subseteq S$

$$(1) \qquad \sum_{u \in X} a'_u \leqslant \sum_{v \in T} \min(b'_v, c(X, \{v\})).$$

In this case, $c(X, \{v\}) = x$ for all $v \in T$, where $x = |X|$. Note that the following bounds hold for a' and b':

$$E(n) \leqslant a'_u \leqslant E(n) + D(n), \quad \forall u \in S,$$

$$E(n) \leqslant b'_v \leqslant E(n) + D(n), \quad \forall v \in T.$$

We now perform a case analysis on the value of x to establish the validity of inequality (1).

Case 1 $[x \leqslant E(n)]$. In this case the inequality (1) is equivalent to the following inequality:

$$\sum_{u \in X} a'_u \leqslant xn.$$

Since $n > E(n) + D(n)$ the above inequality, and hence (1) is valid.

Case 2 $[x > E(n)]$. In this case the inequality (1) is implied by the following inequality:

$$(2) \qquad \sum_{u \in X} a'_u \leqslant \sum_{v \in Y} b'_v + x(n - y), \quad \forall Y \subseteq T, \; y = |Y|.$$

Since each $a'_u \leqslant E(n) + D(n)$, we have that $A_X = \sum_{u \in X} a'_u \leqslant x(E(n) + D(n))$. Clearly, this quantity A_X is less than $x(n - y)$ when $y \leqslant n - E(n) - D(n)$. It follows that the inequality (2) is valid when $y \leqslant n - E(n) - D(n)$. Therefore, we only need to consider the case where $x > E(n)$ and $y \geqslant n - E(n) - D(n)$. Using the bounds on the supplies and demands, we obtain the following inequality.

$$\begin{aligned}\sum_{u \in X} a'_u &= \sum_{u \in S} a'_u - \sum_{u \in S \setminus X} a'_u \\ &\leqslant \sum_{v \in T} b'_v - (n - x)E(n) \\ &\leqslant \sum_{v \in Y} b'_v + \sum_{v \in T \setminus Y} b'_v - (n - x)E(n).\end{aligned}$$

Using $\sum_{v \in T \setminus Y} b'_v \leqslant (n - y)(E(n) + D(n))$, the above inequality implies that the inequality (2) is valid if the following inequality (3) is valid for $n - D(n) - E(n) \leqslant y \leqslant n$.

$$(3) \qquad (n - y)(E(n) + D(n)) - (n - x)E(n) \leqslant x(n - y).$$

If $x > E(n) + D(n)$ then it is easy to see that the inequality (3), and hence (2), must be valid. Therefore, we are only left with the case where $E(n) \leqslant x \leqslant E(n) + D(n)$ and $n - D(n) - E(n) \leqslant y \leqslant n$. Using $x \geqslant E(n)$ and the fact that $n - y \leqslant E(n) + D(n)$, we obtain that the inequality (3) is valid if the following inequality holds:

$$(E(n) + D(n))D(n) \leqslant (n - E(n))E(n).$$

This inequality can easily be verified for our choice of the values of $E(n)$ and $D(n)$. □

We now turn to Step (2) of the algorithm. The next lemma bounds the values of the excess flow sent by the sources and the excess flow received by the sinks. Let $F(n) = D(n) + E(n)$.

LEMMA 3.2. *In the Undirected Transportation Algorithm*

$$\forall_\gamma > 0, \quad \exists l > 0, \quad \text{Prob}\left[\exists u \in S \colon |\bar{a}_u - pa'_u| \geqslant l\sqrt{F(n) \log F(n)}\right] = O(n^{-\gamma}),$$

$$\forall_\gamma > 0, \quad \exists l > 0, \quad \text{Prob}\left[\exists v \in T \colon |\bar{b}_v - pb'_v| \geqslant l\sqrt{F(n) \log F(n)}\right] = O(n^{-\gamma}).$$

PROOF. Let X be the sum of r Bernoulli trials, where each trial assumes the value 1 with probability p and value 0 with probability $1 - p$. The Chernoff bound [Ch] as applied to the tail of binomial distribution states that for any β, such that $0 \leqslant \beta \leqslant 1$,

$$\text{Prob}\left[|X - rp| \geqslant \beta rp\right] \leqslant 2e^{(-\beta^2 rp/3)}.$$

The number of nonzero entries in each row or column of the matrix $M(a', b')$ is at least $E(n)$ and at most $F(n)$. This is the number of trials in a Bernoulli process where each trial is successful with probability $p(n)$. An application of the Chernoff bound for the tail of the binomial distribution completes the proof of the lemma. □

At this stage we are guaranteed that the following bounds hold (with high probability):

$$(4)\quad pE(n) - l\sqrt{F(n)\log F(n)} \leqslant \bar{a}_u - a_u \leqslant pE(n) + l\sqrt{F(n)\log F(n)}, \quad \forall u \in S,$$

$$(5)\quad pE(n) - l\sqrt{F(n)\log F(n)} \leqslant \bar{b}_v - b_v \leqslant pE(n) + l\sqrt{F(n)\log F(n)}, \quad \forall v \in T.$$

Step (3) of the algorithm uses the edges in the backward direction to route the excess flow back from sinks to sources. It is clear that a_u, a'_u, b_v and b'_v are all integral as required for the application of the Fine-Tuning Algorithm. The bounds on the excess, (4) and (5), together with the Fine-Tuning Theorem yield the following lemma.

LEMMA 3.3. *The Fine-Tuning Algorithm succeeds in Step* (3) *with probability* $1 - O(n^{-\alpha})$, *where* $\alpha > 0$ *is a constant which depends on* p.

The probability of failure of each stage of the algorithm is now suitably bounded from above. We now observe that the probability of failure of the entire algorithm is bounded from above by the sum of the probabilities of failure of the various steps in the algorithm even though they may not be independent. This completes the proof of the Undirected Transportation Theorem. Note that we cannot choose $E(n)$ to be larger than $n/\log n$ if we wish to employ the Fine-Tuning Theorem as above. The proof of Lemma 3.1 requires that $E(n)$ should not be much smaller than $D(n)$. This imposes an upper bound of $n/\sqrt{\log n}$ on $D(n)$.

We observe that the Undirected Transportation Algorithm can also be applied to certain cases where $|S| \neq |T|$. The proof of the Undirected Transportation Theorem is easily seen to extend to the case where $|S| \neq |T|$ provided $|S|, |T|$ are large enough, i.e., that they should be substantially larger than $D(n)$ and $E(n)$. Note that to be able to apply the Fine-Tuning Algorithm in the case where $|S| \neq |T|$, we again need that $|S|$ and $|T|$ should be substantially larger than $E(n)$.

3.2. *The undirected max-flow problem.*

We now make use of the Undirected Transportation Theorem to devise an algorithm to solve a probabilistic version of the undirected max-flow problem. Consider an instance of the max-flow problem satisfying the following conditions.

(a) the underlying graph is undirected.

(b) $|S| = |T| = r$ and $|I| = n$, where $r \leqslant n$ may depend on n.

(c) each edge is present with a probability p (a positive constant) and if an edge is present then its capacity is 1.

We assume, without loss of generality, that the $(S, V - S)$ cut has a smaller capacity than the $(V - T, T)$ cut. If this is not the case then we can interchange the roles of the source and sink vertices and reverse the direction of each edge. We present the following theorem:

THEOREM 3.2 (Min-Cut Theorem). *In instances of the probabilistic max-flow problem satisfying conditions* (a)–(c),

$$\text{Prob}[\textit{The } (S, V - S) \textit{ cut is the minimum cut}] = 1 - O(n^{-\alpha})$$

where $\alpha > 0$ *is a constant which depends on* p.

This theorem is proved by presenting a linear time algorithm, called the Max-Flow Algorithm, which constructs a flow saturating the $(S, V - S)$ cut. Clearly, the value of the max-flow is less than or equal to the capacity of this cut.

The Max-Flow Algorithm works in two stages. In the first stage, an instance of the probabilistic transportation problem is created such that a feasible solution to that instance will yield a flow saturating the $(S, V - S)$ cut in the original problem. In the second stage it uses the Undirected Transportation Algorithm to solve the transportation problem instance generated in the first stage.

3.3. *The Max-Flow Algorithm*. Our aim is to find a flow which will saturate the $(S, V - S)$ cut. Clearly, we can ignore edges drawn from $S \times S$ and $T \times T$. Moreover, all edges from $S \times T$ can be saturated (in the forward direction) without affecting the flow through the remaining edges. We now consider only the edges from $S \times I$, $I \times I$ and $I \times T$. The algorithm will find a flow which will saturate all edges from $S \times I$ in the forward direction.

For each $v \in I$ define $\Delta_v = c(S, v) - c(v, T)$, this is the excess of the capacity of incoming edges over the capacity of the outgoing edges. Let $I^+ = \{v \in I: \Delta_v > 0\}$ and $I^- = \{v \in I: \Delta_v < 0\}$. Further define $\Delta^+ = \Sigma_{v \in I^+} \Delta_v$ and similarly $\Delta^- = \Sigma_{v \in I^-} \Delta_v$. We now describe the first stage of the Max-Flow Algorithm.

The Max-Flow Algorithm.

Step (1). Saturate all edges from $S \times I$ by sending flow from sources to intermediate nodes.

Step (2). Saturate all edges from $I \times T$ by sending flow from intermediate nodes to the sinks.

Step (3). Since $c(S, I) \leqslant c(I, T)$ we have $\Delta^+ + \Delta^- \leqslant 0$. At this point, for each $v \in I$, the excess of in-flow over out-flow is exactly Δ_v. Arbitrarily reduce flow along edges drawn from $I^- \times T$ until the net flow across the (S, I) cut equals the net flow across the (I, T) cut. Ensure that, for each $v \in I^-$, the new excess flow, say δ_v, remains nonpositive. Let $\delta^- = \Sigma_{v \in I^-} \delta_v$, then we have $\Delta^+ + \delta^- = 0$ and $\Delta_v \leqslant \delta_v \leqslant 0$, for each $v \in I^-$.

Step (4). Let G^1 be the bipartite subgraph of the original underlying graph G which is induced by the vertex set $S^1 = I^+$ and $T^1 = I^-$. Construct an instance of the capacitated transportation problem with G^1 as the underlying graph, supply Δ_v for $v \in S^1$ and demand $-\delta_v$ for $v \in T^1$.

It is easy to see that the first stage works in linear time. In the second stage of the Max-Flow Algorithm we find a feasible flow for this transportation problem instance using the linear time Undirected Transportation Algorithm described earlier. The final flow is the sum of the two flows constructed in the two stages of the max-flow algorithm. The following theorem results.

THEOREM 3.3 (Max-Flow Theorem). *The Max-Flow Algorithm finds a maximum flow for instances satisfying conditions* (a)–(c) *in linear time with probability* $1 - O(n^{-\alpha})$, *where* $\alpha > 0$ *is a constant which depends on* p.

Observe that the above algorithm will find a flow saturating the $(S, V - S)$ cut if the second stage succeeds. This flow will be a maximum flow. To complete the proof of this theorem we need to show that the transportation problem instance generated by this algorithm satisfies the conditions of the Undirected Transportation Theorem. We first prove the following bound on the size of the excess at each intermediate node. This bound also applies to the size of the supplies and demands of the transportation problem instance generated in the first stage. We also show that $|I^+|$ and $|I^-|$ are both fairly close to $n/2$.

LEMMA 3.4. *In the Max-Flow Algorithm*

$$\forall\gamma > 0, \exists k > 0, \quad \text{Prob}\left[\exists v \in I\colon |\Delta_v| > k\sqrt{n \log n}\right] = O(n^{-\gamma}).$$

PROOF. The result holds for the case where $r \leqslant \sqrt{n}$ since the maximum value of Δ_v is then bounded by $\sqrt{n}$. We now consider the case where $r > \sqrt{n}$.

Consider some $v \in I$. Let X_v denote the number of edges from $S \times \{v\}$ which are actually present in the underlying graph G. Similarly, let Y_v denote the number of edges from $\{v\} \times T$. Clearly, both X_v and Y_v are the sum of r independent Bernoulli trials where each trial assumes value 1 with probability p and value 0 with probability $1 - p$. We make use of the Chernoff bound [Ch] as applied to the tails of binomial distributions. For any β, $0 \leqslant \beta \leqslant 1$, we have

$$\text{Prob}\left[X_v \geqslant (1-\beta)rp\right] \leqslant e^{-\beta^2 rp/2}.$$

Choosing $\beta = \sqrt{(b \log r/r)}$, for some positive constant b, and applying the bound to both X_v and Y_v we have

$$\text{Prob}\left[|\Delta_v| = |X_v - Y_v| \geqslant 2\beta rp\right] \leqslant 4r^{-bp/2}.$$

Summing over all $v \in I$ and choosing $b = 2(1+2\gamma)/p$ and $k = \sqrt{8p(1+2\gamma)}$ we have,

$$\text{Prob}\left[\exists v \in I\colon |\Delta_v| \geqslant k\sqrt{r \log r}\right] = O(r^{-2\gamma}).$$

Since $\sqrt{n} \leqslant r \leqslant n$ we have the desired result. □

LEMMA 3.5. *In the Max-Flow Algorithm*

$$\forall\gamma > 0, \exists f > 0, \quad \text{Prob}\left[\,\big||I^+| - n/2\big| > f\sqrt{n \log n}\right] = O(n^{-\gamma}).$$

PROOF. In our definition of I^+ and I^- we ignored the vertices $v \in I$ for which $\Delta_v = 0$. Since these vertices can be assigned to either set, they can be used to balance the sizes of I^+ and I^-. To simplify the following description we will assume that such vertices will be assigned to either I^+ or I^- equiprobably.

By symmetry, Prob[$v \in I$ is assigned to I^+] = 1/2. Independence follows from the observation that all edge capacities are independently distributed. An application of the Chernoff bound yields the desired result. □

It is clear that the transportation problem instances generated in the first stage satisfy all requirements of the Undirected Transportation Theorem, with one exception. The number of sources (I^+) and sinks (I^-) in the transportation problem instance will not be equal. However, as we remarked earlier, the Undirected Transportation Algorithm can be still be used provided the number of source and sinks is large enough. The bound from Lemma 3.5 shows that this is indeed the case.

4. A directed transportation algorithm.

We now present an algorithm to solve certain instances of the transportation problem where the underlying graph is directed. In particular, we consider instances of the transportation problem satisfying the following conditions.

(a) the underlying graph $G(S \cup T, E)$ is complete and every edge is directed from S to T.

(b) $|S| = |T| = n$.

(c) the edge capacities are i.i.d. random variables drawn from the set $\{0, 1, \ldots K\}$, where $K > 1$ is some constant.

(d) the expected value of the edge capacities is at least $1 + \epsilon$, where ϵ is a positive constant.

(e) the pair (a, b) is $(D + 1)$-realizable, for some integer constant $D > 0$ to be specified later.

Once again our assumptions about the supplies and demands are essentially the weakest possible. Since the expected edge capacity is close to 1, we obviously need that the pair (a, b) is realizable. Our requirement of $(D + 1)$-realizability is only a slight weakening of that necessary condition. We could also have assumed that the supplies and demands are chosen from some reasonable distribution, say a uniform distribution. It is not very hard to see that then the pair (a, b) would have been realizable with high probability. Our result is much stronger because we allow the supplies and demands to be arbitrarily chosen, subject to (e).

We present an algorithm, the Directed Transportation Algorithm, which will solve such instances of the transportation problem with high probability. This leads to the following theorem:

THEOREM 4.1 (Directed Transportation Theorem). *The Directed Transportation Algorithm finds a feasible flow for transportation problem instances satisfying conditions* (a)–(e) *in linear time with probability* $1 - O(n^{-\gamma})$, *for some constant* $\gamma > 0$.

Before we describe the algorithm and prove the Directed Transportation Theorem we present two combinatorial theorems which are useful in the analysis of the Directed Transportation Algorithm.

4.1. *A combinatorial process.* Consider the following combinatorial process. The *state* of the process is an arbitrary placement of n particles, call them $P = \{1, 2, 3, \ldots, n\}$, on integer points of the real line. There may be more than one particle at a given position. The *initial state* has all n particles at the origin. A *state transition* is divided into two distinct steps. Let S be a subset of P such that $|S| = 2k$. The first step in a transition moves every particle in S, for some arbitrary S, one position in the negative direction (say to the left). In the second step, the k leftmost particles are each moved *two* positions to the right (or in the positive direction). It can be shown that no particle will ever move out of the interval $[-A \log n, 2]$, where A is some positive constant.

Let K be a positive integer and θ a positive real number. Consider now the following generalization of the combinatorial process. The definition of the state (as well as the initial state) of the process is as before. The first step of a transition, as before, moves all particles in S, for some arbitrary S, one step each to the left. The second step of the transition involves the choice of n arbitrary integers, $\{d_1, d_2, \ldots, d_n\}$ such that $\sum_{j=1}^{n} d_j = |S|$ and $0 \leqslant d_i \leqslant K$ for each i. We will refer to the requirement that $\sum_{j=1}^{n} d_j = |S|$ as the *balance* constraint. It is also required that $\sum_{j=1}^{t} d_j \geqslant (1 + \theta)t$ for all $t < t_0$, where t_0 is the index of the rightmost nonzero d_i. This last condition will be referred to as the *prefix* constraint. The second step of a transition now moves the tth-leftmost particle d_t positions to the right, for $t = 1, 2, \ldots, t_0$.

The conditions imposed on the second step of a transition ensure that the net rightward movement of a group of t leftmost particles is larger than t. This constraint prevents any particle from straying too far away to the left. The following theorem can be proved about the generalized process.

THEOREM 4.2 (Interval Theorem). *All particles remain in the interval* $[-C \log n, D]$ *of the real line, where* $C = C(K, \theta)$ *and* $D = D(K, \theta)$ *are positive constants.*

The following notation and lemma will be required for the proof of this theorem. Let $p_i(\tau)$ denote the location of the ith particle after τ transitions have taken place. The state of the process at time step τ will be given by the set of locations, $p_i(\tau)$ occupied by the particles i, $1 \leqslant i \leqslant n$. The main tool for the analysis of this combinatorial process will be the following notion of the *moment* at an integer point on the real line.

DEFINITION 4.1. Let p be any integer point on the real line. The *left moment* at p at any time is the sum of the distances from p of all particles to the left of p on the real line:

$$LM(p,\tau) = \sum_i \max(p - p_i(\tau), 0).$$

In the following lemma, we will show that the moment satisfies an invariant inequality at each time step. Using this invariant, we will be able to establish that no particle can move too far away from the origin.

LEMMA 4.1. *There exist A and ρ depending only on θ and K such that*
(a) $A > 0$ *and* $0 < \rho < 1$, *and*
(b) *at each time step τ and for each integer l, $LM(l,\tau) \leqslant nA\rho^{-l}$.*

PROOF. The proof will be by induction on the time step τ. We will assume that the lemma holds for all l at the time step $\tau - 1$, and prove it for all l at time step τ. We will start by proving the induction step, and later show the base case, i.e., at time step 0. The value of the constants ρ and A will also be specified later.

Assume that the left moment at every integer position satisfies the required inequality at time step $\tau - 1$. We now show that it must satisfy the required inequality after the completion of the τth transition, henceforth referred to as the current transition. Observe that it suffices to prove the invariant for a particular position l on the real line, without any loss of generality. Therefore, we are only required to show that $LM(l,\tau) \leqslant nA\rho^{-l}$. The main idea of the proof is to bound the new moment at l by a positive linear combination of the moments at the previous time step. To simplify our notation we will consider everything relative to this location l. We will use the following notation.

NOTATION 4.1. • *Let* cell *i denote the location $l - i$ on the real line, where $i \leqslant K - 1$.*
• *Let* cell *K denote all locations to the left of $l - (K - 1)$.*
• *Let A_i denote the number of particles in* cell *i, before the current transition.*
• *Let a_i denote the number of particles which were moved one unit to the left from cell i in the first stage of the current transition.*
• *Let β_i denote the sum of the distances that particles from cell i were moved to the right in the second stage of the current transition. Thus, $\beta_i = \sum_{j \in S_i} d_j$, where S_i is the set of particles that resided in cell i after the first stage of the current transition.*
• *Let $M(h)$ denote the moment at cell h before the current transition and $M'(h)$ denote the moment at cell h after the current transition. By definition, $M(h) = LM(l - h, \tau - 1)$ and we have to prove that $M'(0) \leqslant nA\rho^{-l}$.*

In general, we will only be interested in particles which lie at cell 0 or to its left. Whenever we refer to a particle at cell i it will be assumed that $0 \leqslant i \leqslant K$, unless otherwise stated.

The moment at cell h, $-1 \leqslant h \leqslant K$, before the current transition is given by the following equation:

$$M(h) = M(K) + \sum_{i=h+1}^{K} (i-h)A_i \leqslant nA\rho^{-l+h}. \tag{6}$$

Let L denote the moment gained at cell 0 during the first stage of the current transition. Also, let R denote the moment lost at cell 0 during the second stage of the current transition. Thus, we have that

$$M'(0) = M(0) + L - R. \tag{7}$$

Observe that each particle to the left of the cell 0 which was moved a unit to the left (in the first stage of the current transition) will contribute to the increase in the moment at cell 0. It is now easy to see that L is given by the following equation:

$$L = \sum_{i=0}^{K} a_i. \tag{8}$$

Consider now a particle which was moved c positions to the right in the second stage of the current transition. It is possible that this particle ended up at a position to the right of cell 0. In that case, its contribution to the decrease in the moment at cell 0 would be less than c. This complicates the computation of the value of R. However, we do know that a particle which was moved from cell i to a position to the right of cell 0 will cause a decrease of i in the moment at cell 0. Further, it is known that a particle can move at most K positions to the right in a single transition. This implies that the particles in cell i (before the second stage of the current transition) must cause a decrease of at least $i/K\beta_i$ in the moment at cell 0. Thus, we have the following lower bound on the value of R:

$$R \geqslant \frac{1}{K} \sum_{i=1}^{K} i\beta_i. \tag{9}$$

It will be convenient to express the above lower bound on R in terms of A_i and a_i. This can be done as follows. Consider the rightmost particle which was moved to the right in the second stage of the current transition. Let t denote the cell to which this particle belonged at the end of the first stage of the current transition. If cell t was to the right of cell 1 then set $t = 1$. It is clear that if $t > 1$ then $\beta_1 = \beta_2 = \cdots = \beta_{t-1} = 0$. The balance constraint on this combinatorical process requires that the net leftward movement in the first stage of any transition be exactly equal to the net rightward movement in the second stage of that transition. Therefore, we have the following inequality:

$$\sum_{i=t}^{K} \beta_i \geqslant \sum_{i=0}^{K} a_i. \tag{10}$$

Note that this inequality need not be tight since there may be particles to the right of cell 0 which were moved to the left in the first stage of the current transition. It is not very hard to see that the inequality also holds in the case where particles to the right of cell 0 were moved rightwards in the second stage.

The number of particles at cell i and to its left after the first stage is $a_{i-1} + \sum_{j=i}^{K} A_j$. We invoke the prefix constraint on the transitions to derive the following inequality for $t + 1 \leqslant i \leqslant K$:

$$\sum_{j=i}^{K} \beta_j \geqslant (1 + \theta)\left(a_{i-1} + \sum_{j=i}^{K} A_j\right). \tag{11}$$

We now use equations (9), (10) and (11) to derive the following lower bound on R, after some algebraic manipulation:

$$R \geqslant \frac{t}{K}\sum_{i=0}^{K} a_i + \frac{1+\theta}{K}\sum_{i=t}^{K-1} a_i + \frac{1+\theta}{K}\sum_{i=t+1}^{K}(i-t)A_i. \tag{12}$$

We are now in a position to give an upper bound on the value of $M'(0)$. Substituting equations (6), (8) and (12) into (7) we obtain the following inequality:

$$M'(0) \leqslant M + \sum_{i=1}^{K} iA_i + \left(1 - \frac{t}{K}\right)\sum_{i=0}^{K} a_i - \frac{1+\theta}{K}\sum_{i=t}^{K-1} a_i - \frac{1+\theta}{K}\sum_{i=t+1}^{K}(i-t)A_i.$$

In order to show that this is bounded by $nA\rho^{-l}$ we have to consider three different cases.

Case I $[t = K]$. In this case the RHS of inequality (13) turns out to be exactly $M(0)$ and, thus, by the induction hypothesis, $M'(0) \leqslant M(0) \leqslant nA\rho^{-l}$.

Case II $[t = K - 1]$. In this case inequality (13) can be seen to imply the following, using the fact that $0 \leqslant a_i \leqslant A_i$:

$$M'(0) \leqslant M(K) + \sum_{i=1}^{K} iA_i + \frac{1}{K}\sum_{i=0}^{K-2} A_i - \frac{\theta}{K}A_K.$$

At this point we invoke equation (6) and make use of the induction hypothesis to obtain the following bound:

$$M'(0) \leqslant nA\rho^{-l-1}\left(\left(1 - \frac{1}{K}\right) + \left(\frac{1-\theta}{K}\right)\rho^K + \frac{\theta}{K}\rho^{K+1}\right) = nA\rho^{-l-1}f(\rho).$$

The last inequality completes the analysis of Case II provided $f(\rho) \leqslant \rho$. It can be shown that this is indeed the case provided ρ is sufficiently close to 1. To see this, it is enough to verify that the function $g(\rho)$,

$$g(\rho) = f(\rho) - \rho = \left(1 - \frac{1}{K}\right) + \left(\frac{1-\theta}{K}\right)\rho^K + \frac{\theta}{K}\rho^{K+1} - \rho,$$

is negative when $\rho < 1$, for ρ close enough to 1. This can be verified by observing that g is continuous at 1, $g(1) = 0$, and $g'(1)$ is positive.

Case III $[1 \leqslant t \leqslant K - 2]$. In this case inequality (13) can be seen to imply the following, using the fact that $a_i \leqslant A_i$:

$$M'(0) \leqslant \left(1 - \frac{1}{K}\right)\left(M(K) + \sum_{i=0}^{K}(i+1)A_i\right)$$
$$+ \left(\frac{1}{K} - \frac{\theta}{K}(K-t)\right)\left(M(K) + A_K\right) + \frac{\theta}{K}(K-t)M(K).$$

At this point we invoke equation (6) and make use of the induction hypothesis to obtain the following bound:

$$M'(0) \leqslant nA\rho^{-l-1}\left(\left(1-\frac{1}{K}\right)+\left(\frac{1}{K}-\frac{\theta}{K}(K-t)\right)\rho^{K}+\frac{\theta}{K}(K-t)\rho^{K+1}\right)$$

$$\leqslant nA\rho^{-l-1}\rho = nA\rho^{-1}.$$

The last inequality also holds when $\rho < 1$, for ρ close enough to 1. This can be verified in the same manner as in Case II.

This concludes the proof of the induction step. Note that ρ is chosen to lie close enough to 1, so as to satisfy all the inequalities derived above. Similarly, A will be chosen such that the base case is satisfied.

Base case. In the initial state all particles were at the origin. This implies that, at time step 0, the left moment at all integer points $p \leqslant 0$ was 0, thus trivially satisfying the invariant. The left moment at location p, for $p > 0$, is simply pn in the initial state. We choose A such that, for all $p > 0$, it is the case that $np \leqslant nA\rho^{-p}$. Clearly, it must be the case that $A \geqslant p\rho^{p}$. Such an A exists since the function $h(p) = p\rho^{p}$ is bounded for $p > 0$. □

We are now ready to prove the Interval Theorem.

PROOF [Interval Theorem]. We first show that no particle can move too far to the left. This may be verified by considering the leftmost point on the real line which has a nonzero moment. Let $p_l < 0$ be the leftmost point on the real line at which a particle may be placed in this combinatorial process. A particle in location p_l contributes 1 to the left moment at $p_l + 1$. Thus, we have the following inequalities:

$$1 \leqslant LM(p_l + 1) \leqslant nA\rho^{-p_l-1}.$$

Therefore, it must be the case that $p_l \geqslant -C\log n$ where $C(K,\theta)$ is a positive constant.

Now we show that a particle cannot move too far to the right. Let $p_r > 0$ be the rightmost position occupied by a particle during this combinatorial process. Consider the first time a particle is moved onto the position p_r. Clearly, this particle must have previously occupied a position at or to the right of $p_r - K$. The prefix condition requires that there be at most $n/(1+\theta)$ particles to the left of a particle which is moved to the right. This implies that there must be at least $\theta n/(1+\theta)$ particles at or to the right of the position $p_r - K$. Therefore, there must be a right moment of at least $(p_r - K)(\theta n/(1+\theta))$ about the origin. It is easy to see that, due to the balance constraint, the right moment must be equal to the left moment at the origin at all times. The left moment at the origin is always less than nA. Thus, we get the following inequality:

$$(p_r - K)\frac{\theta n}{1+\theta} \leqslant nA.$$

This means that $p_r \leqslant D$, where $D(K,\theta)$ is an appropriately chosen constant. □

4.2. *A generalization of the Mendelsohn-Dulmage Theorem.* Mendelsohn and Dulmage [MD] proved the following theorem about bipartite graphs:

THEOREM 4.3 (Mendelsohn-Dulmage Theorem). *Let $G(S \cup T, E)$ be a bipartite graph and let M_1 and M_2 be two matchings in G. Then there exists a matching $M_3 \subseteq M_1 \cup M_2$, such that M_3 covers all the nodes of S covered by M_1 and all the nodes of T covered by M_2.*

The proof of this theorem is constructive and leads to an algorithm which runs in time $O(|S| + |T|)$. We generalize this theorem as follows. Let K be a fixed positive integer. By a K-matrix we mean an $m \times n$ matrix whose entries are drawn from the set $\{0, 1, \ldots, K\}$

THEOREM 4.4 (Generalized Mendelsohn-Dulmage Theorem). *Let M^1 be a K-matrix whose row sums, a_i^1, and column sums, b_j^1, satisfy the following condition:*

$$l_i^s \leqslant a_i^1 \quad \textit{and} \quad b_j^1 \leqslant u_j^t.$$

Similarly, let M^2 be a K-matrix whose row sums, a_i^2, and column sums, b_j^2, satisfy the following condition:

$$a_i^2 \leqslant u_i^s \quad \textit{and} \quad l_j^t \leqslant b_j^2.$$

Then there exists a K-matrix M^3 such that

(a) *the row and column sums of M^3 satisfy the following conditions:*

$$l_i^s \leqslant a_i^3 \leqslant u_i^s, \quad l_j^t \leqslant b_j^3 \leqslant u_j^t.$$

(b) *for each i, j we have $M_{ij}^3 \leqslant \max(M_{ij}^1, M_{ij}^2)$.*

Moreover, a matrix M^3 satisfying the above conditions can be constructed in $O(Kmn)$ time.

PROOF. First, observe that the only case we need to consider is where for each i, j we have $\min(M_{ij}^1, M_{ij}^2) = 0$. If this is not the case then let D be the K-matrix where $\forall i, j,\ D_{ij} = \min(M_{ij}^1, M_{ij}^2)$. Subtract D from both M^1 and M^2. Also, modify the upper and lower bounds on the rows (and columns) of M^1, M^2 by subtracting the row-sums (and column-sums) of the matrix D. Clearly, a solution to the new problem, when added to d, gives a solution to the original problem involving M^1 and M^2.

We now restrict our attention to the case where $\forall i, j,\ \min(M_{ij}^1, M_{ij}^2) = 0$. Let $a_i = \max(a_i^1, a_i^2)$ and $b_i = \max(b_i^1, b_i^2)$, for all i, j. Define a vertex set S such that for each row i there are a_i vertices $s(i, 1), s(i, 2), \ldots, s(i, a_i)$. Similarly, define the vertex set T such that for each column j there are b_j vertices $t(j, 1), t(j, 2), \ldots, t(j, b_j)$. We will construct matchings X^1 and X^2 on the bipartite vertex set $S \cup T$ corresponding to the two matrices M^1 and M^2, respectively. We describe the construction of the matching X^1 only. The other matching, X^2, can be constructed analogously.

For each nonzero entry M_{ij}^1, introduce M_{ij}^1 edges into X^1 connecting vertices corresponding to row i with vertices corresponding to column j. It is easy to ensure that each vertex in $S \cup T$ has at most one edge incident on it. This process yields a matching of cardinality $\sum_{i=1}^m a_i^1 = \sum_{j=1}^n b_j^1$.

We now invoke the Mendelsohn-Dulmage Theorem to construct a third matching, $X^3 \subseteq X^1 \cup X^2$, which covers all the vertices in S covered by X^1 and those in T covered by X^2. To derive the matrix M^3 from the matching X^3, set the value of M_{ij}^3 to be the number of edges in M^3 which connect vertices corresponding to row i with those corresponding to column j.

Clearly, the row-sums of M^3 respect the upper bounds since the number of vertices corresponding to each row do so. The row-sums of M^3 are seen to respect the lower bounds since the matching X^3 covers at least as many vertices of each row as the row-sums of M^1. Similar reasoning shows that the column-sums of M^3 respect both the upper and lower bounds. This establishes condition (a) of the theorem.

To verify the validity of condition (b), recall that $\min(M_{ij}^1, M_{ij}^2) = 0$. This implies that the number of edges in $X^1 \cup X^2$ connecting vertices corresponding to a row i

with the vertices corresponding to a column i is less than $\max(M_{ij}^1, M_{ij}^2)$. Thus, the value of M_{ij}^3 cannot exceed $\max(M_{ij}^1, M_{ij}^2)$.

Finally, note that the entire proof is constructive. Moreover, the construction described above requires only $O(Kmn)$ time. This concludes the proof. □

4.3. *The Directed Transportation Algorithm.* The key idea behind the Directed Transportation Algorithm can also be formulated in terms of the mimicking paradigm. We first set aside a small fraction of the edge capacities for the purposes of the Fine-Tuning Algorithm. Next, we construct the $(D+1)$-realization of the supply/demand vectors. This corresponds to the solution of the deterministic relaxation of the probabilistic transportation problem. The solution to the relaxed problem is then mimicked to obtain a partial solution to the original problem. The mimicking process is considerably more sophisticated than that used for the transportation problem. Finally, we use the reserved capacity to fine-tune the solution to obtain a feasible flow.

We now present a brief outline of the mimicking process used by this algorithm. The mimicking process works in a row-by-row fashion, i.e., the algorithm computes the flow matrix for the probabilistic instance by mimicking in order the rows of the flow matrix for the deterministic instance. It is ensures that the row-sums of the solution created by this process are exactly equal to the desired values. Consider the stage where the first $i-1$ rows have already been mimicked. This means that we have created a partial flow matrix for which the entries of the first $i-1$ rows have already been determined. We now describe how the entries of the ith row will be computed. At this point there may be a discrepancy in the column-sums (for the first $i-1$ rows) between the deterministic and the mimicking solutions. Let P_j denote the excess of the jth partial column-sum in the mimicking solution over that in the deterministic solution. While determining the values for the ith row we will consider the columns in increasing order of discrepancy. The edges corresponding to the entries in row i are saturated until the desired row-sum is achieved. The behavior of the column discrepancies is analogous to the combinatorial process outlined earlier. To make the analogy complete it will be necessary to introduce a certain amount of fictitious capacity, as will be explained later. Let $\theta = 1/N$, where N is a fixed positive integer such that $0 < \theta < \epsilon$. Also, let D be the constant $D(K, \theta)$ determined by the Interval Theorem.

The Directed Transportation Algorithm.

Step (1). Set aside a small fraction of the edge capacities for use by the Fine-Tuning Algorithm. For each edge with a nonzero capacity, $\hat{c}(i,j)$, set aside one unit of capacity with probability δ independent of the other edges, where $0 < \delta < \epsilon - \theta$. Let the expected value of the new capacities, $\{c(i,j)\}$, be $1+\epsilon'$; then $\theta < \epsilon'$.

Step (2) [*Deterministic Relaxation*]. Let $a = (a_i)$ and $b = (b_j)$ denote the supply and demand vectors, respectively. In linear time, construct the $(D+1)$-realization of the pair (a, b). Let $M = M(a', b')$ be the resulting matrix.

Step (3) [*Mimicking Process*]. Construct a flow $X = (x_{ij})$, row-by-row, such that $\sum_{j=1}^{n} x_{ij} = a_i'$ (for each i) and $\sum_{r=1}^{i} x_{rj} \approx \sum_{r=1}^{i} M_{rj}$ (for each j). The flow X can be constructed as follows. Suppose we are currently processing row i. Let $P_j = \sum_{r=1}^{i-1} x_{rj} - \sum_{r=1}^{i-1} M_{rj}$, for each column j, where $P_j = 0$ if $i = 1$.

Step (3.1). $P_j \leftarrow P_j - M_{ij}$, for each j.

Step (3.2). Let $c_1, c_2, \ldots, c_n$ denote the capacities of edges going from source i to the sinks in increasing order of P_j. Increase these capacities from c_l to c_l' so as to ensure that $\sum_{l=1}^{t} c_l' \geqslant (1+\theta)t$ and that $c_k \in [0, K]$, for each t, k. This may be done by choosing c_l' to be $\max\{c_l, [(1+\theta)l - \sum_{k=1}^{l-1} c_k']\}$. The extra capacity introduced in this fashion will be called the *fictitious* capacity. Let U_i denote the amount of fictitious capacity required for row i; i.e., $U_i = \Sigma_l(c_l' - c_l)$.

Step (3.3). Send out a_i' units of flow out of source i by considering the edges in increasing order of P_j. Send out c_j' units of flow along the jth such edge, until a total of a_i' units of flow have been shipped out. It may be observed that, due to the introduction of the fictitious capacities, the flow along an edge may exceed the actual capacity. Let x_{ij} denote the flow sent along the edge (i, j).

Step (3.4). $P_j \leftarrow P_j + x_{ij}$, for each j.

Step (4). Repeat the mimicking process of Step (3) with the roles of the rows and columns interchanged. In other words, construct a flow by mimicking the deterministic solution in a column-by-column fashion, using row discrepancies and introducing fictitious capacities as in Step (3). Let Y be the flow obtained in this manner. Also, let V_j denote the total amount of fictitious capacity introduced in column j.

Step (5). Construct a flow X' which satisfies all capacity constraints by appropriately reducing the flow X along edges with fictitious capacities. Similarly, construct a flow Y' from the flow Y.

Step (6). Consider the two K-matrices X' and Y'. Using the Generalized Mendelsohn-Dulmage algorithm, compute a third K-matrix Z such that the row- and column-sums of Z satisfy the bounds satisfied by the row- and column-sums of X' and Y'. The Generalized Mendelsohn-Dulmage algorithm ensures that each entry of Z matrix is no more than the larger of the corresponding entries in the X' and Y' matrices and hence no more than the corresponding edge's capacity.

Step (7) [*Fine-Tuning*]. Let $\bar{a}$ and $\bar{b}$ denote the row and column sum vectors for the flow matrix Z. In Step (1) the capacity set aside for each edge (i, j) was $\hat{c}(i, j) - c(i, j)$. Using these capacities and the Fine-Tuning Algorithm, construct a flow Z' with supply and demand vectors $(a - \bar{a})$ and $(b - \bar{b})$, respectively. The sum of the two flows, Z and Z', is a feasible flow for the transportation problem instance under consideration.

4.4. *Analysis of the Directed Transportation Algorithm*. The proof of the Directed Transportation Theorem will be presented via the following lemmata. But, first, observe that the algorithm runs in linear time since each step requires at most $O(n^2)$ operations, and the size of input is $\Omega(n^2)$. We now proceed to show that each step of the algorithm succeeds with high probability.

LEMMA 4.2. *At the end of Step* (1), *the remaining capacities have expected value at least* $1 + \epsilon' = 1 + \epsilon - \delta$.

PROOF. Consider an edge (i, j). The original capacity of this edge is $c(i, j)$ and the capacity at the end of Step (1) is $\hat{c}(i, j)$. Let p_k = Prob[edge (i, j) has $c(i, j) = k$], for each $k \in [0, K]$. Then we have the following bound:

$$\operatorname{Exp}[\hat{c}(i,j)] = \sum_{k=1}^{K} p_k[\delta(k-1) + (1-\delta)k] \geqslant \operatorname{Exp}[c(i,j)] - \delta \geqslant 1 + \epsilon - \delta.$$

□

A generalized random walk analysis yields the following bounds on the amount of fictitious capacity introduced by the algorithm.

LEMMA 4.3. *For all* $\gamma > 0$, *there exists* $s > 0$ *such that* $\max_i U_i \leqslant s \log n$ *with probability* $1 - O(n^{-\gamma})$.

PROOF. We make use of the analysis of a generalized one-dimensional random walk [Fe] to prove this result. Consider the following random walk process. The particle is initially at some integral position $z > 0$. The rth step is given by the random variable X_r which takes only integral values. Let S_r denote the position of the particle after r steps. Then $S_0 = z$ and $S_t = \Sigma_{r=1}^{t} X_r + z$ for $t > 0$. Let $u_z(a)$ denote the probability of the particle going to a position $\leqslant 0$ before it goes to a position $\geqslant a$, for some fixed integer $a > z$.

Suppose the following conditions hold:

(a) (X_r) are i.i.d. random variables.

(b) $m = \text{Exp}[X_1] > 0$.

(c) $X_r \in [-\nu, \mu]$, where ν, μ are positive integers.

It can be shown [Fe, pp. 363–366] that

$$u_z(z) \leqslant \frac{\sigma^{a+\mu-1} - \sigma^z}{\sigma^{a+\mu-1} - 1},$$

where σ is the unique positive root (other than 1) of the characteristic function of the probability distribution of X_r. It can be shown that $0 < \sigma < 1$ when $m > 0$. Note that σ is a constant which only depends on the distribution of (X_r).

Consider now the fictitious capacity introduced for row i in Step (3.2). Let $U_i(t) = (1 + \theta)t - \Sigma_{r=1}^{t} c_r$, then $U_i = \max_t U_i(t)$. Recall that $\theta = 1/N$ where N is a positive integer such that $0 < 1/N < \epsilon'$. We want to show that U_i cannot be too large.

Let us relate the fictitious capacity to the random walk as follows. Let $X_r = Nc_r - (N + 1)$. We now have that $S_t = -NU_i(t) + z$; here z is the initial position for the random walk which will be specified later. Since the random variables (c_r) are drawn from the set $\{0, 1, 2, \ldots, K\}$ and $\text{Exp}[c_r] = 1 + \epsilon'$, we have

(a) $X_r \in \{Nd - (N + 1) \colon d \in \{0, 1, 2, \ldots, K\}\}$.

(b) $\text{Exp}[X_r] = \epsilon' N - 1 > 0$.

(c) $\nu = -(N + 1)$, $\mu = N(K - 1) - 1$.

Suppose we select $a = n + z$ and $z = k \log n$, for some constant $k > 0$. Now, $u_z(a)$ was defined as being the probability that the value of S_t falls below 0 before it has ever risen above a. Equivalently, it is the probability that the value of U_i rises above z/N before it has ever fallen below $-(a - z)/N$. Clearly, the value of U_i can never fall below $-(a - z)/N$ unless $\Sigma_{r=1}^{t} c_r \geqslant n \geqslant a_i'$. The processing of the row is over if n units of flow have been shipped out. This establishes that $u_z(a)$ is the probability that U_i is greater than $k \log n/N$. We now have

$$\text{Prob}\left[U_i > \frac{k}{N} \log n\right] = u_z(a) \leqslant \frac{\sigma^{a+\mu-1} - \sigma^z}{\sigma^{a+\mu-1} - 1} \leqslant \sigma^z.$$

Summing the probability of failure over all n rows and given that $0 < \sigma < 1$, we have for some $\gamma > 0$,

$$\text{Prob}\left[\max_i U_i \leqslant \frac{k}{N} \log n\right] = 1 - O(n^{-\gamma}). \quad \square$$

The proof of the next lemma is identical.

LEMMA 4.4. *For all* $\gamma > 0$, *there exists* $s > 0$ *such that* $\max_j V_j \leqslant s \log n$ *with probability* $1 - O(n^{-\gamma})$.

Consider now the flow X constructed in Step (4). Since we used a certain amount of fictitious capacity, the *actual* flow X' is slightly less than X. The next lemma gives bounds on the amount of actual flow leaving a source as well as the actual flow entering a sink.

LEMMA 4.5. *Consider the flow matrix X'. The ith row-sum of X' (the net flow out of source i) is $a_i - U_i - (D+1)$, while the jth column sum of X' (the net flow into sink j) is less than $b_j - 1$.*

PROOF. We draw an analogy between the construction of the flow X and the combinatorial process described earlier. The value of P_j, after the processing of row $i-1$, corresponds to the position of particle j after $i-1$ state transitions. The set $\{j|M_i j = 1\}$ corresponds to the set S arbitrarily chosen for the ith state transition. The flow x_{ij} routed to the column j with the tth smallest P_j corresponds to the distance d_t moved by the tth leftmost particle. The use of fictitious capacities ensures that, for each t, $\sum_{k=1}^{t} c'_k \geqslant (1+\theta)t$ as required by the combinatorial process.

It follows that the net actual flow out of source i is exactly $a'_i - U_i = a_i - U_i - (D+1)$. We know that the net flow sent into sink j is $b'_j + P_j$. Invoking the Interval Theorem, we have that $P_j \leqslant D$. It follows that the net actual flow into sink j is no more than $b'_j + D = b_j - 1$. □

The proof of the following lemma is similar to that of Lemma 4.5.

LEMMA 4.6. *Consider the flow matrix Y'. The ith row-sum of Y' (the net flow out of source i) is at most $a_i - 1$, while the jth column sum of Y' (the net flow into sink j) is exactly $b'_j - V_j = b_j - V_j - (D+1)$.*

Lemma 4.5, Lemma 4.6 and the Generalized Mendelsohn-Dulmage Theorem together give us the following lemma.

LEMMA 4.7. *Consider the flow matrix $Z(\bar{a}, \bar{b})$ constructed in Step (6). It satisfies the following conditions:*

$$a_i - U_i - (D+1) \leqslant \bar{a}_i \leqslant a_i - 1, \quad \forall i.$$

$$b_j - V_j - (D+1) \leqslant \bar{b}_j \leqslant b_j - 1, \quad \forall j.$$

To complete the proof of the Directed Transportation Theorem we need to show that Step (7) succeeds with high probability.

LEMMA 4.8. *The Fine-Tuning algorithm succeeds in Step (7) with probability $1 - O(n^{-\gamma})$, for some constant $\gamma > 0$.*

PROOF. From Lemma 4.3, Lemma 4.4 and Lemma 4.7 we have

$$1 \leqslant a_i - a'_i = O(\log n), \quad \forall i,$$

$$1 \leqslant b_j - b'_j = O(\log n), \quad \forall j.$$

In Step (1) we set aside a certain fraction of the edge capacities. The reserved capacity for an edge (i, j) is 1 with probability at least δ/K, which is fixed independent of n; the reserved capacity of an edge is 0 with probability at most $1 - (\delta/K)$. Moreover, the algorithm sets aside the capacities independently for each edge. For each supply or demand in the residual problem, the lower bound is $L(n) = 1$ and the upper bound is $U(n) = O(\log n)$ It is clear that the residual problem at this stage completely satisfies the requirements of the Fine-Tuning Theorem. The Fine-Tuning Algorithm is now applicable to the supplies $a - \bar{a}$ and the demands $b - \bar{b}$. □

4.5. *Application to the directed max-flow problem.* We now apply the Directed Transportation Theorem to the solution of a probabilistic version of the directed max-flow problem. Consider the instances of the max-flow problem where the underlying graph is directed and the following conditions are satisfied:

(a) $S = \{s\}$, $T = \{t\}$ and $I = \{1, 2, \dots, n\}$.

(b) $\forall i \in I$, $c(s, i) = a_i$ and $c(i, t) = b_j$.

(c) $\forall i, j \in I$, where $i \neq j$, the capacities $c(i, j)$ are i.i.d. random variables drawn from the set $\{0, 1, 2, \dots, K\}$ with expectation at least $1 + \epsilon$, where $\epsilon > 0$.

(d) The pair (a, b) is $(D + 1, \infty)$-realizable for some constant $D > 0$.

Observe that our assumption (d) is weaker than assuming that (a, b) is $(D + 1)$-realizable. We will exhibit a linear time reduction from such instances of the max-flow problem to instances of the directed transportation problem satisfying the conditions for the Directed Transportation Theorem. This leads to a linear time algorithm for the max-flow problem and proves the following,

THEOREM 4.5 (Directed Max-Flow Min-Cut Theorem). *Let I be an instance of the directed max-flow problem satisfying conditions* (a)–(d). *With high probability, the cut consisting of all edges leading out of the source (or into the sink) is a minimum cut. Moreover, there exists a linear time algorithm to find a maximum flow in I, with high probability of success.*

We now specify the linear time reduction which will prove the above theorem. Let I be an instance of the max-flow problem satisfying conditions (a)–(d). We will reduce I to an instance C of the directed transportation problem. Construct a directed bipartite graph B as follows. Let $S' = \{s_1, s_2, \dots, s_n\}$ and $T' = \{t_1, t_2, \dots, t_n\}$ denote the bipartite vertex set for B. Associate with each source vertex s_i the supply a_i and with each sink vertex t_j associate the demand b_j. Every edge (s_i, t_j) is present and all edges are directed from S' to T'. Let $c(s_i, t_j) = c(i, j)$ for $i \neq j$. We choose $c(s_i, t_i) = \infty$, for each i. It is now easy to see that a feasible flow for C can always be transformed (in linear time) to a feasible flow for I which saturates all edges leading out of s. Clearly, this would be a maximum flow for I.

To apply the Directed Transportation Algorithm to the instance C, we must make a small modification in Step (2). We first construct a $(D + 1, \infty)$-realization of (a, b) instead of the $(D + 1)$-realization. Notice that this does not affect the rest of the algorithm or its analysis, provided we ignore the "diagonal" edges in the remaining steps of the algorithm. The Directed Transportation Theorem, when applied to C, implies that C is almost surely feasible. Since the Directed Transportation Algorithm will almost surely find a feasible flow for C, we can now derive, in linear time, a maximum flow for I.

5. Further work.

Generalizations of current results. The results presented above could be extended in many directions. It would be interesting to consider different distributions for the edge capacities. We could also look at the case of sparse graphs, i.e., graphs where the probability of an edge, $p(n)$, is small. Another possibility is to consider instances of the directed transportation problem where $K = K(n)$ and $\epsilon = \epsilon(n)$, where the former goes to infinity with n while the latter goes to zero as n approaches infinity.

Large diameter graphs. It has been empirically observed that most network flow algorithms tend to have their worst performance on graphs of large diameter. It would be interesting to consider flows on random graphs which have large diameters. This could be done by considering sparse graphs or random *layered* graphs where the diameter can be increased by increasing the number of layers of vertices.

Mimicking deterministic solutions. It should be possible to apply this technique to other problems. In particular, we might consider probabilistic instances of *Multi-commodity Flow* problems as a natural extension of the problems considered above. Another possibility is the *Minimum-Cost Flow* problem. Of course, there is no reason why this technique should work only for flow problems.

Acknowledgements. Dr. Karp was supported by NSF Grant DCR-8411954.

Part of this work was done when Rajeev Motwani was at the Computer Science Division, University of California, Berkeley and supported by NSF Grant DCR-8411954.

Noam Nisan was supported by NSF Grant DCR-8411954. Part of this work was done when he was at the Computer Science Division, University of California, Berkeley.

References

[Al] Alon, N. (1990). Generating Pseudo-Random Permutations and Maximum Flow Algorithms. *Informat. Process. Lett.* **35** 201–204.

[AV] Angluin, D. and Valiant, L. (1979). Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings. *J. Comput. and System Sci.* **19** 155–193.

[Bo] Bollobás, B. (1985). *Random Graphs*. Academic Press, New York.

[CH] Cheriyan, J. and Hagerup, T. (1989). A Randomized Maximum-Flow Algorithm. *Proc. 30th Annual Sympos. Foundations of Computer Sci.* 118–123.

[Ch] Chernoff, H. (1952). A Measure of Asymptotic Efficiency for Tests Based on the Sum of Observations. *Ann. Math. Statist.* **23** 493–509.

[Di] Dinic, E. A. (1970). Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation. *Soviet Math. Dokl.* **11** 1277–1280.

[DJ] Doulliez, P. and Jamoulle, E. (1972). Transportation Networks with Random Arc Capacities. *Rev. Francaise Automat. Informat. Recherche Oper.* **3** 45–60.

[ER1] Erdös, P. and Rényi, A. (1959). On Random Graphs. *Publ. Math. Debrecen*, **6** 290–297.

[ER2] _____ and _____ (1964). On Random Matrices. *Publ. Math. Inst. Hungar. Acad. Sci.* **8** 455–461.

[ER3] _____ and _____ (1968). On Random Matrices. II. *Studia Sci. Math. Hungar.* **3** 459–464.

[Fe] Feller, W. *An Introduction to Probability Theory and Its Applications. Vol* I. John Wiley & Sons, New York.

[FoFu] Ford, L. R. and Fulkerson, D. R. (1962). *Flows in Networks*. Princeton University Press, Princeton, NJ.

[FrFr] Frank, H. and Frisch, I. T. (1971). *Communication, Transmission and Transportation Networks.* Addison-Wesley, Reading, MA.

[FrHa] _____ and Hakimi, S. L. (1965). Probabilistic Flows through a Communication Network. *IEEE Trans. Circuit Theory*, **CT-12** 413–414.

[FR] Fulkerson, D. R. and Ryser, H. J. (1961). Widths and Heights of (0, 1)-Matrices. *Canad. J. Math.* **13** 239–255.

[Ga] Gale, D. (1957). A Theorem on Flows in Networks. *Pacific J. Math.* **7** 1073–1082.

[GT] Goldberg, A. V. and Tarjan, R. E. (1986). A New Approach to the Maximum Flow Problem. *Proc. 18th ACM Sympos. Theory of Computing*. 136–146.

[GH] Goldschmidt, O. and Hochbaum, D. S. (1990). A fast perfect-matching algorithm in random graphs, *SIAM Journal on Discrete Mathematics*, **3** 48–57.

[GS] Grimmett, G. R. and Suen, H. S. (1982). The Maximal Flow through a Directed Graph with Random Capacities. *Stochastics* **8** p. 153–159.

[GW] _____ and Welsh, D. R. A. (1982). Flow in Networks with Random Capacities. *Stochastics*, **7** 205–229.

[HZ] Hassin, R. and Zemel, E. (1988). Probabilistic Analysis of the Capacitated Transportation Problem. *Math. Oper. Res.* **13** 80–89.

[Ho] Hochbaum, Dorit S. (1988). An Exact Sublinear Algorithm for the Max-Flow, Vertex-Disjoint Paths, and Communication Problems on Random Graphs. Preliminary Draft.

[Ka] Karp, R. M. (1979). The Probabilistic Analysis of Combinatorial Optimization Algorithms. *Tenth Internat. Sympos. Math. Programming*.

[KLMR] Karp. R. M., Lenstra, J. K., McDiarmid, C. J. H. and Rinnooy Kan, A. H. G. (1985). Probabilistic Analysis. *Combinatorial Optimization: Annotated Bibliographies*. (Ed.) M. O'hEigeartaigh, J. K. Lenstra and A. H. G. Rinnooy Kan, John Wiley & Sons, New York, 52–88.

[La] Lawler, E. L. (1976). *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York.

[MD] Mendelsohn, N. S. and Dulmage, A. L. (1958). Some Generalizations of the Problem of Distinct Representatives. *Canad. J. Math.* **10** 230–241.

[Mo1] Motwani, R. (1988). Probabilistic Analysis of Matching and Network Flow Algorithms. Ph. D. Thesis. University of California at Berkeley.

[Mo2] ______. (1989). Expanding Graphs and Average-Case Analysis of Algorithms for Matchings and Related Problems. *Proc. 21st ACM Sympos. Theory of Computing*, 550–561.

[Mo3] ______. (1990). The Realization of 0/1 Matrices with Unbounded Diagonals. Preliminary Draft.

[PB] Prékopa, A. and Boros, E. (1986). On the Probability of the Existence of a Feasible Flow in a Transportation Network. RUTCOR Research Report No. 20-86, The State University of New Jersey, Rutgers.

[Ry] Ryser, H. J. (1957). Combinatorial Properties of Matrices of Zeros and Ones. *Canad. J. Math.* **9** 371–377.

R. M. Karp: Computer Science Division, University of California, Berkeley, California 94720

R. Motwani: Department of Computer Science, Stanford University, Stanford, California 94305

N. Nisan: Hebrew University of Jerusalem, Jerusalem, Israel