アルゴリズム　23－5
(1991. 9. 19)

ヒッチコック輸送問題の新算法

徳山 豪, 中野 淳

日本ＩＢＭ東京基礎研究所

あらまし　n個のソースとk個のシンクを持つヒッチコック型輸送問題において，nがkに比べて十分に大きい場合の解法について考える．この問題はすでに知られている最小費用流のアルゴリズムを用いると$O(n^2k\log n + n^2\log^2 n)$時間で解くことができるが，$n > k\log k$の時には$O(k^2n\log^2 n)$に改善できることを示す．さらに*splitter finding*と呼ばれる幾何学的手法とランダマイゼーションにより，最大供給量と最小供給量の比cが$\frac{n}{k^4\log n}$より小さければ上記の計算時間をさらに短くすることができる．特に$c \le \frac{n}{k^7\log^7 n}$の場合には問題はオプティマルである$O(kn)$時間で解ける．

Efficient Algorithms for the Hitchcock Transportation Problem

Takeshi Tokuyama and Jun Nakano

IBM Research, Tokyo Research Laboratory
5-11, Sanbancho, Chiyodaku, Tokyo 102

Abstract　We consider the Hitchcock transportation problem on n supply points and k demand points when n is much greater than k. The problem is solved in $O(n^2k\log n + n^2\log^2 n)$ time if we directly apply an efficient minimum-cost flow algorithm. We show that the complexity can be improved to $O(k^2n\log^2 n)$ time if $n > k\log k$. Further, applying a geometric method named *splitter finding* and randomization, we improve the time complexity for a case in which the ratio c of the least supply and the maximum supply is bounded by $\frac{n}{k^4\log n}$. Indeed, if $c \le \frac{n}{k^7\log^7 n}$, the problem is solved in $O(kn)$ time, which is optimal.

1 Introduction

Imagine a distributed data system with k strage devices $D_1, D_2, \ldots, D_k$ and n data $\{z_1, z_2, \ldots, z_n\}$. The size of D_i is λ_i, and the size of z_j is ω_j. Each data is called by processors, and if the data z_j is placed in D_i the (expected) communication time for calling a unit of the data z_j is known to be $\alpha_{i,j}$. The problem of how to find the allocation of data that minimizes the communication cost is a linear programming problem called the Hitchcock transportation problem. Its standard form is as follows:

$$\text{Minimize} \quad \sum_{i=1}^{k}\sum_{j=1}^{n} \alpha_{i,j} y_{i,j} \tag{1}$$

subject to

$$\sum_{j=1}^{n} y_{i,j} = \lambda_i \quad (i = 1, 2, \ldots, k) \tag{2}$$

$$\sum_{i=1}^{k} y_{i,j} = \omega_j \quad (j = 1, 2, \ldots, n) \tag{3}$$

$$y_{i,j} \geq 0 \quad (i = 1, 2, \ldots, k;\ j = 1, 2, \ldots, n) \tag{4}$$

The problem is feasible if and only if $\sum_{i=1}^{k} \lambda_i = \sum_{j=1}^{n} \omega_j$. Because of the symmetry of the problem, we can assume $n \geq k$ without loss of generality.

We can relax this feasibility condition to $\sum_{i=1}^{k} \lambda_i \leq \sum_{j=1}^{n} \omega_j$ and replace (2) by the inequality

$$\sum_{j=1}^{n} y_{i,j} \geq \lambda_i \quad (i = 1, 2, \ldots, k),$$

since by defining $\lambda_{k+1} = \sum_{j=1}^{n} \omega_j - \sum_{i=1}^{k} \lambda_i$, and $\alpha_{k+1,j} = 1$ for $j = 1, 2, \ldots, n$, the relaxed problem is transformed into the standard form.

The Hitchcock transportation problem is a kind of capacitated minimum-cost flow problem, which is known to be soluble in strongly polynomial time [9, 3, 4, 8]. Indeed, we can transform the problem into the minimum-cost flow problem on a network with $N = n + k$ nodes and $M = kn$ edges. The best known algorithm [O] for solving the uncapacitated minimum-cost flow problem is $O(N \log N(M + N \log N))$, which solves the Hitchcock transportation problem in $O(kn^2 \log n + n^2 \log^2 n)$ time if $n \geq k$. This complexity is more than the square of n even if k is very small.

In real life, the number n of supply points is often much larger than the number k of demand points (or vice versa). Indeed, in the example shown above, the number of data is usually much greater than the number of devices. Matsui [6] gives a linear-time solution to the Hitchcock transportation problem with respect to n if k can be considered as a constant. However, the time complexity of his algorithm is $O(n(k!)^2)$ if k is not a constant; thus it is too expensive unless k is extremely small.

If $\omega_j = 1$ for any $j = 1, 2, \ldots, n$, the problem is named the λ-assignment problem. Tokuyama and Nakano [10] give a geometric approach named *splitter finding* to the λ-assignment problem, and solve it in $O(nk + n^{0.5}k^{3.5})$ randomized time (this result has been recently improved to $O(nk + n^{0.5}k^{2.5} \log n)$ by the authors [11]).

In this paper, we first give a geometric interpretation of the problem, and show that Orlin's algorithm can solve the Hitchcock transportation problem in $O(k^2 n \log^2 n)$ time if it is efficiently implemented. Furthermore, if there is a constant c such that $1 \leq \omega_j \leq c$ for $1 \leq j \leq n$, we give a randomized algorithm, which runs in $O(nk + c^{1/3}n^{2/3}k^{10/3} \log^{7/3} n)$ time with high probability. If $n > ck^7 \log^7 ck$, the running time is $O(nk)$, which is optimal.

2 Geometric interpretation

Given the Hitchcock transportation problem defined by (1),...,(4), we consider the following geometric model. For each index $j (j = 1, 2, \ldots, n)$, we define a point $p(j) = (\alpha_{1,j}, \ldots, \alpha_{k,j})$ in k-dimensional real space R^k.

Let $S = \{p(1), p(2), \ldots, p(n)\}$. We call ω_j the weight of $p(j)$ (which is often called the supply of $p(j)$ in literature).

Let $G = (g_1, g_2, \ldots, g_k)$ be a point on the hyperplane L defined by the equation $x_1 + x_2 + \cdots + x_k = 0$. The region satisfying $x_i - g_i \leq x_h - g_h$ for any $h \neq i$ is called the i-th (closed) region split by G. We denote this region by $T(G; i)$. The space subdivision into $T(G; i)$ $(i = 1, 2, \ldots, k)$ is called a *splitting* (Figure 1). G is called a *splitter.*

Theorem 1. (Existence of λ-Splitter) *There exists a splitter G such that the sum of the weights of the points of S in $\cup_{i \in J} T(G; i)$ is greater than or equal to $\sum_{i \in J} \lambda_i$ for all $J \subset \{1, 2, \ldots, k\}$.*

The splitter G defined in the theorem is called a λ-*splitter* of the point set S. The corresponding splitting is called a λ-splitting, which is a generalization of that given in [10].

Theorem 2. *For a λ-splitter G, the Hitchcock transportation problem has a solution satisfying $y_{i,j} = \omega_j$ if $p(j)$ is in the interior of $T(G; i)$, and $y_{i,j} = 0$ if $p(j)$ is outside $T(G; i)$. Conversely, any solution of the Hitchcock transportation problem satisfies the above condition for a λ-splitter.*

Proof. It is easy to see that, if we divide the boundary elements of regions suitably, we can find a solution of (2), (3), and (4) that satisfies the conditions of the theorem. We will show that this solution minimizes the total cost. We define $\beta_{i,j} = \alpha_{i,j} - g_i$ for $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, n$. Then, the optimal solution does not change if we replace $\alpha_{i,j}$ by $\beta_{i,j}$ for all i and j. In the new problem, the splitter is the origin O. The region $T(O; i)$ is defined by $x_i \leq x_h$ for each $h \neq i$. Thus, it is clear that the above solution minimizes the total cost. We omit the proof of the converse statement. □

This theorem implies that if we find a splitter, we can solve the Hitchcock transportation problem by finding the transportation of the points on the boundaries between regions. We call a λ-splitting a λ-transportation if we have already assigned the weights of the boundary elements.

Each region of a splitting has $k - 1$ boundary facets. Thus, there are $k(k-1)/2$ facets in total. We say that S is simple if the number of incidence relations between the boundary facets and points is at most $k - 1$ for any splitter. Since the degree of freedom of G is $k-1$, we can always make S simple by giving a small perturbation. More precisely, we may use the SOS system of Edelsbrunner [1]. Further, the algorithm in this paper can be easily applied to a non-simple case almost directly without loss of time complexity. Thus, we assume from now on that S is simple.

Although the splitter is not unique, the following fact is observed:

Proposition 3. *The solution of the Hitchcock transportation problem is unique if S is simple.*

3 Scaling method

In this section, we give an efficient algorithm based on the minimum-cost flow algorithm of Orlin [8]. Our algorithm is an incremental algorithm that uses the scaling method. The main theorem in this section is as follows:

Theorem 4. *The Hitchcock transportation problem is soluble in $O(k^2 n \log^2 n)$ time.*

The rest of this section is devoted to proving the theorem. In this section, we assume that the weights are integers. The maximum weight is Λ, and let $l = \lfloor \log \Lambda \rfloor$. First, we design an $O(k^2 n l \log n)$ time algorithm, which resembles the Edmonds-Karp's algorithm [2].

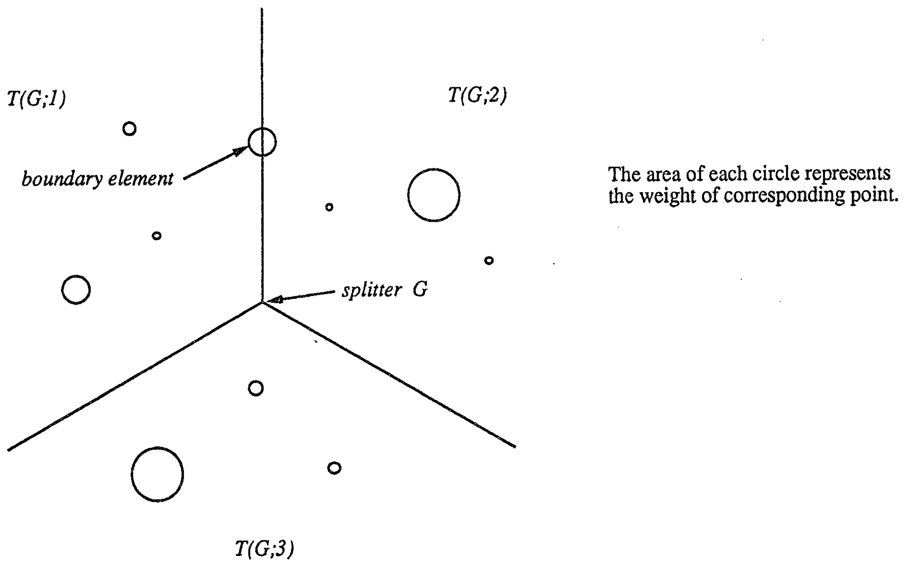
T(G;1)
T(G;2)
boundary element
splitter G
T(G;3)
The area of each circle represents the weight of corresponding point.

Figure 1: Splitter and splitting of point set (projected on L)

We decompose a point p at a maximum of l points associated with the 2-adic decomposition of the weight. The created point is denoted by $p^{\{i\}}$ if it has the weight 2^i. Thus, we have a set $\tilde{S}$ of at most nl points, each of which has a weight that is a power of 2.

For a number a and an integer j, we define the value $a^{(j)} = 2^j \lfloor a/2^j \rfloor$. We define $\lambda^{(j)} = (\lambda_1^{(j)}, \dots, \lambda_k^{(j)})$.

The algorithm is an incremental algorithm, which is divided into l stages.

Procedure Hitchcock

```
1.    $G =$ origin;
2.    $N = 0$;  { $N$ is the total weight of points
                inserted so far}
3.    for $j = 0$ to $l$;
      begin
3.1.      Split($j, G, N$);
      end;
end;
```

Procedure Split(j, G, N)

```
1.    $L =$ the list of all points of weight $2^{l-j}$;
2.    while $L$ is not empty, and $N < \sum_{i=1}^{k} \lambda_i^{(l-j)}$;
      begin
2.1.      choose a point $p$ from $L$;
2.2.      Add $p$ in the region (say, $T(G;s)$) containing
          $p$ in the current splitting;
2.3.      if the total weight assigned to $T(G;s)$
          exceeds $\lambda_s^{(l-j)}$ then
2.3.1         Update($G$);
          end if;
2.4.      $N = N + 2^{l-j}$;
      end while;
3.    split each remaining point in $L$ into two points
      of weight $2^{l-j-1}$;
4.    return;
end;
```

The subroutine **Update**(G) updates the current transportation so that no region overflows.

The current splitter is $G = (g_1, g_2, \ldots, g_k)$. The current total weight assigned to $T(G;i)$ is denoted by γ_i. Thus, G is a γ-splitter of the set of currently existing points. We are at the j-th stage, a new point is inserted into $T(G;s)$, and **Update** is called. Without loss of generality, we assume at least one point of $\tilde{S}$ is assigned to each region. We say that an element of $T(G;i)$ is a *proper* element if a portion of its weight is assigned to $T(G;i)$ in the transportation. Clearly, an interior element is a proper element. We make the following assumptions:

Assumption 1: There exists a region $T(G;t)$ such that $\gamma_t \leq \lambda_t^{(l-j)} - 2^{l-j}$.

Assumption 2: For each proper element of each region, the portion of its weight assigned to the region is an integer multiple of 2^{l-j}.

For each ordered pair of two regions $T(G;a)$ and $T(G;b)$, we compute the point $q(a,b)$, which is the nearest proper element in $T(G;a)$ to the boundary between two regions. $q(a,b)$ may be a boundary element. We construct a complete directed graph $\mathcal{G}$ on the node set $\{v_1, v_2, \ldots, v_k\}$. For the directed edge (v_a, v_b), we give a cost $q(a,b)_b - q(a,b)_a$, where $q(a,b)_i$ is the i-th coordinate value of $q(a,b)$. It is easy to see that there is no negative cost cycle in this graph.

We compute the minimum-cost path from v_s to all other nodes, and make the union of the minimum-cost paths, which becomes a directed tree called the minimum-cost path tree rooted by v_s. Then, the new splitter $G' = (g'_1, \ldots, g'_k)$, which by definition satisfies $\sum_{i=1}^{k} g'_i = 0$, is defined by the following equations:

$$g'_a - g'_b = q(a,b)_a - q(a,b)_b \qquad (5)$$

for all pairs (a,b) such that (v_a, v_b) is a directed edge in the minimum-cost path tree.

Since there are $k-1$ adjacent relations in the tree, G' is uniquely determined. The following is the key lemma:

Lemma 5. *A proper element of $T(G;i)$ is in $T(G';i)$.*

Proof. As in Figure 2, we can find G' by translating the boundary facets without going beyond a point. We omit the details in this version. □

Corollary 6. *G' is a γ-splitter of the point set before insertion.*

We will show there is a transportation after the insertion so that the weight assigned to $T(G;t)$ is $\gamma_t + 2^{l-j}$ for a fixed region satisfying Assumption 1, and that assigned to $T(G;i)$ is γ_i for $i \neq t$.

We consider the minimum cost path from v_s to v_t, and push a flow of size 2^{l-j} along it. From the definition of G', if the flow is pushed from v_a to the adjacent node v_b, there exists a proper boundary element of $T(G';a)$ on the boundary between $T(G';b)$. From Assumption 2, we can move 2^{l-j} of the weight of the boundary element from $T(G';a)$ to $T(G';b)$. Thus, we can push the weight on the splitting so that the weight of $T(G;s)$ (resp. $T(G;t)$) is diminished (resp. increased) by 2^{l-j}.

Hence, we can update the transportation so that no region overflows under Assumptions 1 and 2. It is easy to see that the update procedure preserves Assumptions 1 and 2. Moreover, if these assumptions hold for one stage, then they clearly hold for the next stage. Since the assumptions hold initially, we can assume these assumptions throughout the algorithm. Now, it is easy to verify that the algorithm correctly solves the transportation problem.

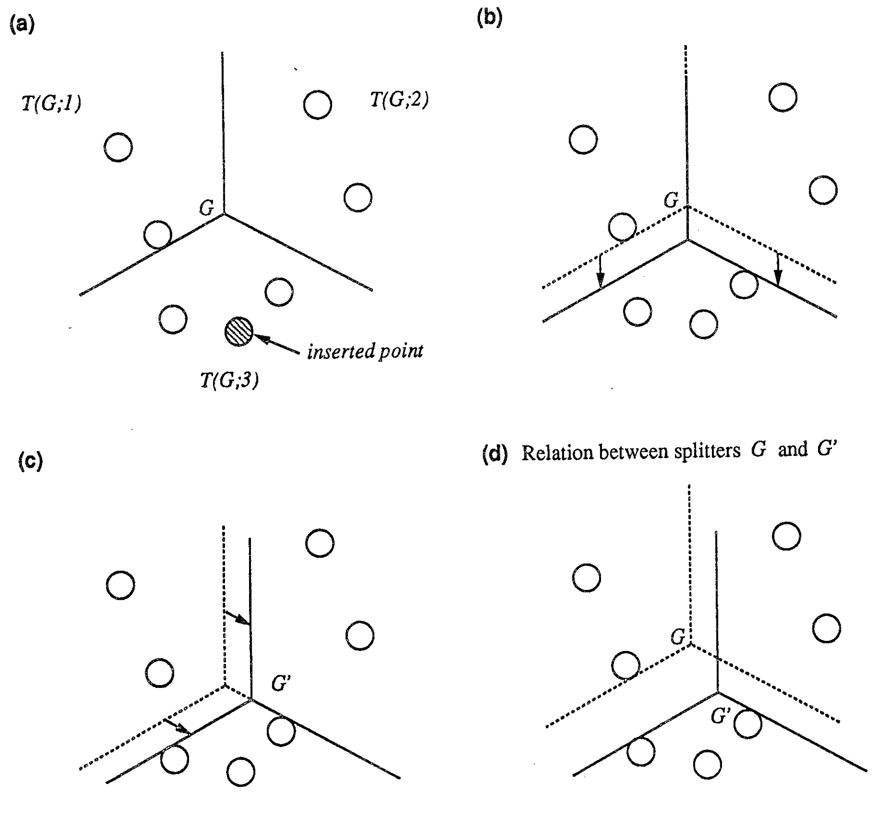

Figure 2: Update of the splitter

Proposition 7. *The scaling algorithm solves the Hitchcock transportation problem in $O(k^2 nl \log n)$ time.*

Proof. Let $G = (g_1, g_2, \ldots, g_k)$ be the current splitter. When a point $p = (p_1, p_2, \ldots, p_k)$ is inserted, we find the index h such that $p_h - g_h = \min_i\{p_i - g_i\}$. Then, it is easy to see that p is contained in $T(G; h)$. This operation requires $O(k)$ time. When **Update** is called, we must find $k(k-1)$ points $\{q(a,b)_a, q(a,b)_b | a, b = 1, 2, \ldots, k\}$. If we provide $k(k-1)$ priority queues, these points can be found in $O(k^2 \log n)$ time. The storage needed for these priority queues is $O(kn)$, and thus is asymptotically not more than the input size. The time complexity for computing the minimum-cost path tree of $\mathcal{G}$ is $O(k^2 \log k)$. Thus, the overall complexity for an insertion is $O(k^2 \log n)$. During each stage, at most k points are split. Thus, the total number of insertions is not more than $nl + kl$. Thus, the proposition is proven. □

The above algorithm is not strongly polynomial. In order to make it strongly polynomial, the technique of Orlin [8] is applied. We consider a point p in S with a weight ω. If $\omega = \sum_{j \in J} 2^j$, p is decomposed into $\{p^{(j)} | j \in J\}$ of $\tilde{S}$. Suppose the first $1 + \lceil \log nk \rceil$ largest portions of p have already been inserted.

The size of each inserted point during the current stage is at most $\frac{1}{2nk}\omega$. Thus, the total weights of the remaining points of $\tilde{S}$ is at most $n\{\frac{\omega}{4nk}(1 + 1/2 + 1/4 + \cdots)\} < \frac{\omega}{2k}$ after this stage has been completed.

On the other hand, at least $(1 - \frac{1}{2nk})\omega$ of the weight of p has been inserted. Thus, at the end of the current stage, there exists a region (which we denote as the $f(p)$-th region) to which at least $\frac{1}{2k}\omega$ of the weight of p is assigned. Thus, the portion of p assigned in the i-th region is greater than the total weight of the remaining points, and we can show the following easily:

Claim. In the final splitting, p is in the $f(p)$-th region.

Thus, we modify the algorithm as follows: If the first $1 + \lceil \log nk \rceil$ portion of a point p has been inserted, we insert all the remaining portions just after the current stage.

In this case, the Assumption 2 fails. However, the following Assumption 2′ holds:

Assumption 2′: In the j-th step, for each proper element of each region, its weight assigned to the region is more than $2^{(l-j)}$.

It is easy to see that the insertion procedure works if we replace Assumption 2 by Assumption 2′. Thus, we can reduce the number of insertions from $O(nl)$ to $O(n \log nk)$. We have Theorem 4.

We remark that the expected performance of the algorithm would be much better than its worst case complexity, since the number of updates of the splitter is usually much smaller than the number of insertions. An insertion without updating the splitter is done in $O(k)$ amortized time, if we use Fibonacci heaps as priority queues in our date structure.

4 Randomized algorithms for the c-grain case

The scaling algorithm solves the Hitchcock transportation problem in $O(k^2 n \log^2 n)$ time. However, in the special case in which $\omega_i = 1$ for $i = 1, 2, \ldots, n$, the problem is known to be solved in $O(kn + k^{2.5} n^{0.5} \log n)$ time by a randomized algorithm [10, 11].

Let us try to use randomized method under more relaxed conditions.

Definition. A Hitchcock transportation problem is called c-grain for a constant $c \geq 1$ if $1 \leq \omega_i \leq c$ for

$1 \le i \le n$. Here, ω_i are real numbers.

In this section, we show that we can improve the performance of the algorithm by pruning points by using random sampling for a c-grain transportation problem, if $n > ck^4 \log ck$.

Theorem 8. *The c-grain Hitchcock transportation problem is solved in $O(nk + c^{1/3}n^{2/3}k^{10/3}\log^{7/3} n)$ time with probability larger than $1 - \frac{1}{n^\gamma}$ for any constant γ.*

In particular, if $n > ck^7 \log^7 ck$, the problem is solved in the optimal $O(nk)$ time.
We remark that we can often decompose some large-weighted points into smaller pieces to reduce c. Further, we will see later that we don't mind if there exist small number of points whose weights are less than 1.

Now we give an algorithm. Let us randomly choose a subset S_0 consisting of m points of S. Let G be the λ-splitter that we would like to find.

Lemma 9. *The total weight of the points of S_0 located in $T(G;j)$ is less than $\frac{m\lambda_j}{n} + \sqrt{\frac{rcm\lambda_j}{n}}$ with probability more than $1 - e^{-r}$*

Proof. The random variable X corresponds to the sum of the weights of points of S_0 located in $T(G;j)$. We define the random variable $X(p)$ as follows: If the point p is in $T(G;j)$, then $X(p) = \omega(p)$ (the weight of the point), otherwise $X(p) = 0$. Thus, the expected value of X is

$$E(X) = \sum_{p\in S} E(X(p)) = m\left(\frac{1}{n}\sum_{p\in T(G;j)} \omega(p)\right) = \frac{m\lambda_j}{n}.$$

Here, for a c-grain case, the following Chernoff-like bounds hold:

$$\Pr\Big(X > (1+\delta)E(X)\Big) < \exp\left\{-\frac{E(X)\delta^2}{4c}\right\}.$$

$$\Pr\Big(X < (1-\delta)E(X)\Big) < \exp\left\{-\frac{E(X)\delta^2}{4c}\right\}.$$

The lemma is easily derived from them. □

We define a vector $\mu(1) = (\mu(1)_1, \dots, \mu(1)_k)$ as follows: $\mu(1)_j = \frac{m\lambda_j}{n} + \sqrt{\frac{rcm\lambda_j}{n}}$ for $j \ne 1$, and $\mu(1)_1 = m - \sum_{i=2}^{k} \mu(1)_j$. Then, we find the $\mu(1)$-splitter $G(1)$ of S_0. We can show the following lemma:

Lemma 10. *The probability that $G(1)$ is located in $T(G;1)$ is greater than $1 - e^{-r}$.*

Proof. If $G(1)$ is located in the interior of $T(G;j)$ for $j \ne 1$, then all points in $T(G(1);j)$ are in $T(G;j)$. Thus, there are more than $\mu(1)_j$ points in $T(G;j)$, which can occur with a probability of less than e^{-r}, because of the previous lemma. □

Corollary 11. *The probability that all the points of S in the interior of $T(G(1);1)$ lie in the interior of $T(G;1)$ is greater than $1 - e^{-r}$.*

Now, let $S(1)$ be the set of points of S located in $T(G(1);1)$ (Figure 3). Analogously, we define $G(i)$, $\mu(i)$, and $S(i)$ for $i = 1, 2, \dots, k$.

Let ν_i be the total weight of points of $S(i)$. The expected value of ν_1 is $\frac{n}{m}\mu_1$, and ν_1 is more than

$$\begin{aligned}\frac{n}{m}\left(\mu(1)_1 - \sqrt{\frac{crm\lambda_1}{n}}\right) &= \lambda_1 - \sum_{i=1}^{k}\sqrt{\frac{crn\lambda_i}{m}} \\ &\le \lambda_1 - n\sqrt{\frac{crk}{m}}\end{aligned}$$

with a probability greater than $1 - e^{-r}$. Similarly, we can estimate ν_i for $i = 2, \dots, k$.

Our randomized algorithm is as follows: We choose a sample of size m, set $r = \log k + \gamma \log n$ for a constant γ, and find $G(1), \dots, G(k)$.

For each point p of S, we find the region containing it in the splitting with respect to $G(1)$. If it happens to be in $T(G(1);1)$, p is a point in $S(1)$. If it is in $T(G(1);i)$ for an $i \ne 1$, we check whether it is contained in $T(G(i);i)$.

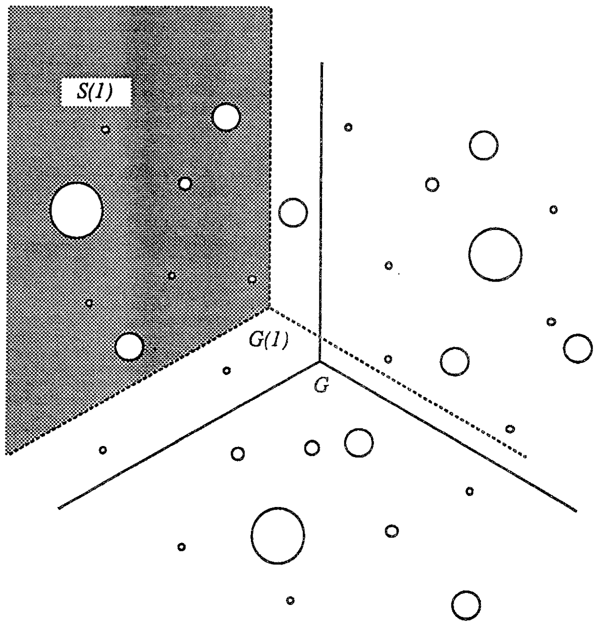

Figure 3: The set $S(1)$

This operation needs $O(k)$ time for each point. Thus, we find a subset $S'(i)$ of $S(i)$. It is easy to show that there is no point of S_0 in $T(G(i);i) - T(G(1);i)$. Thus, from the ϵ-net theory [5], the number of points in $\bigcup_{i=2}^k(S(i) - S'(i))$ is $O(\frac{rkn\log k}{m})$ with probability $1 - e^{-r}$. Thus, we can find almost all points of $\bigcup_{i=1}^k S(i)$.

Now we define the set $\bar{S} = S - \bigcup_{i=1}^k S'(i)$. Let ν_i' be the total weight of points in $S'(i)$, and let $\xi = \lambda - \nu'$. Then,

Proposition 12. *A ξ-splitter of $\bar{S}$ is a λ-splitter of S with probability $1 - ke^{-r}$.*

Thus, we can find the λ-splitter of S by computing the ξ-splitter of $\bar{S}$ by applying the scaling algorithm.

Let us analyze the complexity of the algorithm. The probability $1 - ke^{-r}$ is $1 - \frac{1}{n^\gamma}$, since we chose $r = \log k + \gamma\log n$. Since the weight of points in $S - \bigcup_{i=1}^k S(i)$ is $kn\sqrt{\frac{crk}{m}}$, the number of points there is also bounded by $kn\sqrt{\frac{crk}{m}} + \frac{rkn\log n}{m}$. Thus, it costs $O(k^2\log^2 n\{kn(\frac{r\log k}{m} + \sqrt{\frac{ckr}{m}}\})$ time to compute the splitter of $\bar{S}$. On the other hand, using the scaling algorithm to compute the splitters of the sample, it costs $O(k^3m\log^2 m)$ time. If we set $m = c^{1/3}r^{1/3}k^{4/3}n^{2/3}$, we obtain the Theorem 8.

This algorithm is a Monte-Carlo-type algorithm, since it may output a wrong answer. However, we can make it Las-Vegas. The following is the key lemma.

Lemma 13. *Suppose G be a ν-splitter of a point set S. We insert a point p of weight ω. Let λ satisfy that $\lambda_i \geq \nu_i$ and $\sum_{i=1}^k \lambda_i = \omega + \sum_{i=1}^k \nu_i$. Then, if no point of S has a weight less than ω, we can find a λ-splitter in $O(k^3\log^2 k + k^2\log n)$ time, if we permit $O(nk\log n)$ preprocessing time.*

We omit the proof of this lemma in this version. Using this lemma, we can design an $O(k^3 n \log^2 k + k^2 n \log n)$ time algorithm by inserting points in the largest-first fashion.

Moreover, for the c-grain case, even if the output of the randomized algorithm is not a correct splitter, we can delete a small number of points from the overflow regions and re-insert these points. When we insert a point of size a, we decompose it into $\lceil a \rceil$ points of size less than or equal to 1, and apply Lemma 13. Then a point is inserted in $O(ck^3 \log^2 k + ck^2 \log n)$ time, and we can obtain the correct transportation in a small update time plus the preprocessing time. We can reduce the preprocessing time to $O(nk)$ in this case (we omit the details). Thus, we obtain a Las-Vegas algorithm.

Furthermore, we can generalize Theorem 8 as follows:

Theorem 14. *If there are $h = o(n)$ points each of whose weight is less than 1, and other $n - h$ points satisfy the c-grain condition, the Hitchcock transportation problem is solved in $O(nk + c^{1/3} n^{2/3} k^{10/3} \log^{7/3} n + h\{k^3 \log^2 k + k^2 \log n\})$ time with probability larger than $1 - \frac{1}{n^\gamma}$ for any constant γ.*

References

[1] Edelsbrunner, H., Algorithms in Combinatorial Geometry, ETACS Monograph in Theoretical Computer Science 10, Springer Verlag (1986).

[2] Edmonds, J., and Karp, R. M., Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems, *J. ACM 19,* pp.248–264, (1972).

[3] Fujishige, S., An $O(m^3 \log n)$ Capacity-rounding Algorithm for the Minimum Cost Circulation Problem: A Dual Framework to Tardos's Algorithm, *Math. Prog. 35,* pp.298–309, (1986).

[4] Galil, Z. and Tardos, E., An $O(n^2(m + n \log n) \log n)$ Min-cost Flow Algorithm. *Proc. 27th IEEE FOCS* pp.136–146, (1986).

[5] Haussler, D. and Welzl, E., ϵ-nets and simplex range queries, *Disc. Comp. Geom. 2* pp.127–151, (1987).

[6] Matsui, T. Linear Time Algorithm for Hitchcock Transportation problem with Fixed Number of Supply Points, *Preprint* (1991).

[7] Numata, K., and Tokuyama, T., Splitting a Configuration in a Simplex, Lecture Notes in Computer Science 450, pp.429–438, Springer-Verlag (1990).

[8] Orlin, J., A Faster Strongly Polynomial Minimum Cost Algorithm, Proc. 20th ACM STOC, pp.377–387, (1988).

[9] Tardos, E., A Strongly Polynomial Minimum Cost Circulation Algorithm, Combinatorica 5, pp.247–255 (1985).

[10] Tokuyama, T. and Nakano, J., Geometric Algorithms on an Assignment Problem, *Proc. 7th ACM Symposium on Computational Geometry* pp.262–271 (1991).

[11] Tokuyama, T. and Nakano, J., Unpublished result.