アルゴリズム 21－7
（1991. 5. 29）

割り当て問題に対する幾何的算法

徳山 豪，中野 淳
日本アイ・ビー・エム東京基礎研究所

完全二部グラフ ($\Gamma = (A, B)$, $|A| = n$, $|B| = k$) における最小コスト 1 対多マッチング問題をそれと同等な幾何学的問題 (最小コスト λ 割り当て問題) におきかえ，randomization のテクニックを用いることにより，$O(kn + k^{3.5}n^{0.5})$ 時間のアルゴリズムを発見した．これは，$k \leq n^{0.6}$ がなり立つ時には，すでに知られている $O(kn^2 + n^2 \log n)$ 時間のアルゴリズムを凌ぐものである．

Geometric algorithms for a minimum cost assignment problem

Takeshi Tokuyama and Jun Nakano
IBM Research, Tokyo Research Laboratory,
5-11 Sanbancho, Chiyoda-ku, Tokyo 102, Japan.

We consider the minimum cost λ-assignment problem, which is equivalent to the minimum weight one-to-many matching problem in a complete bipartite graph $\Gamma = (A, B)$, where A and B have n and k nodes respectively. Formulating the problem as a geometric problem, we give an $O(kn + k^{3.5}n^{0.5})$-time randomized algorithm, which is better than existing $O(kn^2 + n^2 \log n)$-time algorithm if $k \leq n^{0.6}$.

1 Introduction

We consider the following transportation problem named the *minimum cost λ-assignment problem*: Suppose a company has n employees and k sites. For each $i = 1, 2, \ldots, k$, the i-th site needs λ_i workers, satisfying $\sum_{i=1}^{k} \lambda_i \leq n$. The commutation cost for the employee u to the site v is defined by $\omega(u, v)$. The problem is to minimize the total commutation cost.

We can assume $\sum_{i=1}^{k} \lambda_i = n$ without loss of generality, since we can consider an imaginary site with capacity $n - \sum_{i=1}^{k} \lambda_i$ to which any employee can commute with cost zero. The problem can be mathematically formulated as follows:

Let $\Gamma = (A, B)$ be a complete bipartite graph, where A has n nodes $u_1, u_2, \ldots, u_n$ and B has k nodes $v_1, v_2, \ldots, v_k$. We assume that $n \geq k$. A real number $\omega(u, v)$, called the weight of $e(u, v)$, is associated with each edge $e(u, v)$. A *capacity vector* is a sequence $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k)$ of nonnegative integers of length k such that $\sum_{i=1}^{k} \lambda_i = n$.

Definition 1. (Figure 1) Given a capacity vector λ, a subgraph H of Γ is called a λ-assignment if the node set of H is $A \cup B$, the degree of each node of A is 1, and the degree of v_i is λ_i for each $i = 1, 2, \ldots, k$.

Definition 2. The minimum cost λ-assignment is the λ-assignment H that minimizes the total cost $\sum_{(u,v) \in H} \omega(u, v)$.

Problem 1. Find the minimum cost λ-assignment.

Problem 1 can be transformed into a minimum cost flow problem, and solved in $O(kn^2 + n^2 \log n)$ time by Fredman and Tarjan's algorithm [FT], and in $O(kn^{1.5} \log nL)$ time by Gabow and Tarjan's scaling algorithm [GT], where $\log L$ is the bit length to represent weights. However, if the order of magnitude of k is much smaller than that of n, which often occurs in practical applications, it seems that the study of efficient algorithms has not been well done. For instance, a comparatively naive $O(k^3 n + k^2 n \log n)$ time algorithm shown in Section 2 often outperforms above sophisticated algorithms.

In this paper, the problem is formulated as a geometric problem named *splitter finding*, and an efficient randomized algorithm, which terminates in $O(kn + k^{3.5} n^{0.5})$ time with probability $1 - e^{-cn^{0.2}}$ (for any constant c), is given. This algorithm is faster than the Fredman-Tarjan's algorithm and the Gabow-Tarjan's algorithm if $k < n^{0.6}$ and $k < (n \log nL)^{0.4}$, respectively. Further, if k can be considered as a constant, an $O(n)$ time deterministic algorithm is given. Table 1 shows the time complexities of strongly polynomial algorithms to solve the minimum cost λ-assignment problem depending on the relation between n and k (c_0 is a suitable constant).

2 A geometric interpretation

The minimum cost λ-assignment problem is equivalent to a geometric problem named the *λ-splitter finding*, essentially defined by Numata and Tokuyama [NT]. For each node u of A, we associate the vector $p(u) = (\tilde{\omega}(u, v_i))_{i=1,2,\ldots,k}$, where $\tilde{\omega}(u, v_i) = \omega(u, v_i) - \frac{1}{k} \sum_{j=1}^{k} \omega(u, v_j)$.

Thus we obtain a set S of n points on the hyperplane $L(0) : x_1 + x_2 + \cdots + x_k = 0$ in the real k-dimensional space $\mathbf{R}^k$. For a given point $G = (g_1, g_2, \ldots, g_k)$ in $L(0)$, the half space defined by $x_i - x_j \leq g_i - g_j$ is denoted by $\mathcal{H}(G; i, j)$. The (closed) region $T(G; i) = \bigcap_j \mathcal{H}(G; i, j)$ is called the i-th region split by G.

Let λ be a capacity vector (which is called an ordered partition of n in [NT]). A partition of the point set S into $\{S_i\}_{i=1,2,\ldots,k}$ is called a *λ-partition*

k	deterministic	randomized
$k \geq n^{0.6}$	$O(kn^2)$	—
$k \leq n^{0.6}$	$O(kn^2)$	$O(k^{3.5} n^{0.5})$
$k \leq n^{0.5}$	$O(k^3 n)$	$O(k^{3.5} n^{0.5})$
$k \leq n^{0.2}$	$O(k^3 n)$	$O(kn)$
$k \leq \log n$	$O(k^2 n \log n)$	$O(kn)$
$k \leq \frac{c_0 \log \log n}{\log \log \log n}$	$O((k!)^2 n)$	$O(kn)$

Table 1: Time complexity

if S_i contains exactly λ_i points. Obviously, there is a natural one-to-one correspondence between the set of λ-partitions and that of λ-assignments.

A λ-partition is called a λ-splitting if there exists a point G in $L(0)$ such that $S_i \subset T(G;i)$ for each $i = 1, 2, \ldots, k$. The point G is called the λ-splitter (Figure 2).

Theorem 1. *A λ-partition corresponds to a minimum cost λ-assignment if and only if it is a λ-splitting.*

Proof. Let $\{S_i\}$ be a λ-partition corresponding to a minimum cost assignment. We fix a λ-splitter G. Without loss of generality, we can assume there is no element of S located on the boundary face of the splitting. Assume there is a point $r(i,j) \in S_i$ in the region $T(G;j)$ such that $j \neq i$. If we denote the s-th coordinate value of $r(i,j)$ by $r(i,j)_s$, $r(i,j)_j - r(i,j)_i < g_j - g_i$ by definition of the splitter. We write $i = \sigma_1, j = \sigma_2$. Since $T(G;\sigma_2)$ contains λ_{σ_2} elements, there is at least one element (denoted by $r(\sigma_2, \sigma_3)$) of S_{σ_2} which is located in $T(G;\sigma_3)$ for a suitable $\sigma_3 \neq \sigma_2$. If we continue this chain (pruning it if necessary), we find a cycle $r(\sigma_1,\sigma_2), r(\sigma_2,\sigma_3), \ldots, r(\sigma_{t-1},\sigma_t)$ of length $t \leq k$, satisfying $\sigma_t = \sigma_1$. If we re-assign the point $r(\sigma_i, \sigma_{i+1})$ for the $S_{\sigma_{i+1}}$ for each $i = 1, 2, \ldots, t-1$, we obtain a new λ-partition. Since

$$\sum_{i=1}^{t-1} \left\{ r(\sigma_i,\sigma_{i+1})_{\sigma_{i+1}} - r(\sigma_i,\sigma_{i+1})_{\sigma_i} \right\} < \sum_{i=1}^{t-1} (g_{\sigma_{i+1}} - g_{\sigma_i}) = 0,$$

the corresponding new assignment has a smaller weight, which contradicts the hypothesis.

It is easier to show that a λ-splitting always makes a minimum cost λ-assignment. In particular, λ-splitting is unique if the point set satisfies certain non-degeneracy conditions [NT]. ∎

We present an incremental algorithm for finding a λ-splitter. Suppose we have already inserted m points of S. The set of inserted points is denoted by $\tilde{S}$. We consider a partition $\mu = (\mu_1, \ldots, \mu_k)$ of m satisfying the condition that $\mu_i = \lfloor \frac{m}{n}\lambda_i \rfloor$ (called a *short part*) or $\lfloor \frac{m}{n}\lambda_i \rfloor + 1$. Assume that we have computed a μ-splitter G_0 of $\tilde{S}$. $\tilde{S}_i$ denotes the set

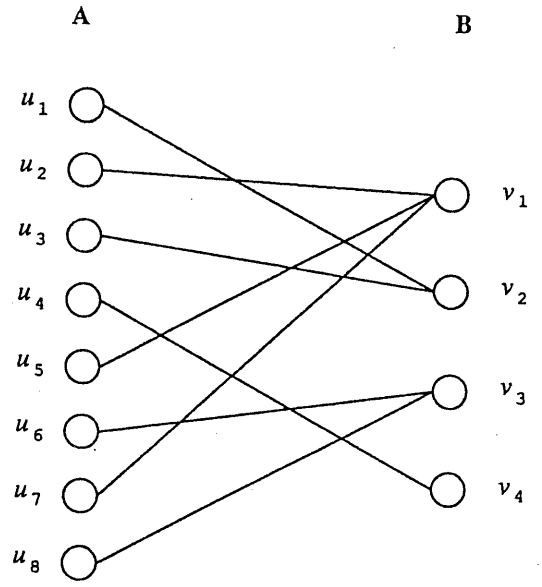

Figure 1: (3,2,2,1)-assignment

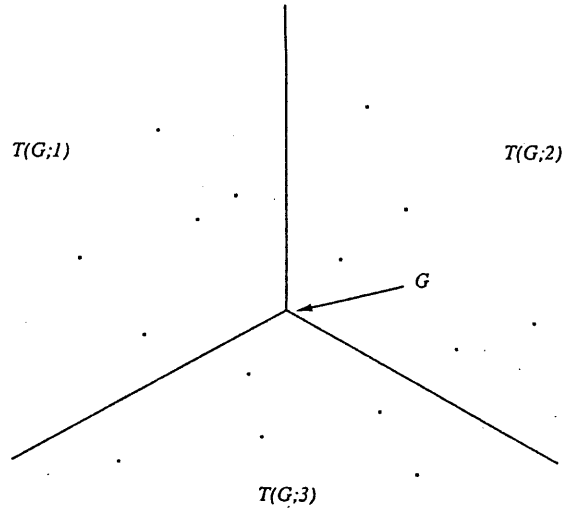

Figure 2: (5,5,5)-splitter

of points of $\tilde{S}$ assigned to the i-th region $T(G_0; i)$. $p(i,j)$ $(j \neq i)$ is defined by the point of $\tilde{S}_i$ satisfying $p(i,j)_i - p(i,j)_j = \max_{x \in \tilde{S}_i}\{x_i - x_j\}$. We use certain priority queues to maintain these $k(k-1)$ points. When we insert a new point p, we first determine in which region of this splitting the point is located.

Lemma 1. *The region containing p is determined in $O(k)$ time.*

Proof. Compute the vector $p - G_0$, and find the coordinate i with the minimum value. Then, p is contained in the i-th region $T(G_0; i)$. This operation obviously takes $O(k)$ time. ∎

Next, we add the point p to the corresponding part, say, $\tilde{S}_a$. If μ_a is a short part, then we can add another new point without updating the splitter. Otherwise, the part contains $\lfloor \frac{m+1}{n}\lambda_a \rfloor + 2$ points, and there must be at least one short part. We update the splitter to reduce the number of points in $\tilde{S}_a$ by 1, and increase the number of points of one of the short parts. The following is the algorithm for updating the splitter (Figure 3 on the next page): Let J be a subset of $K = \{1, 2, \ldots, k\}$. $\bar{X}(J)$ is the vector in $\mathbf{R}^k$ such that its i-th component is 1 if $i \in J$ and 0 otherwise. By projecting $\bar{X}(J)$ on the hyperplane $L(0)$, we obtain a vector $X(J)$. Initially, $J = \{a\}$. We consider the line $\ell(J)$ passing through G_0 with the direction vector $X(J)$. For each point $p(i,j), i \in J$ and $j \in K - J$, we consider the intersection $g(i,j)$ of the hyperplane $x_i - x_j = p(i,j)_i - p(i,j)_j$ and $\ell(J)$. We replace G_0 by the nearest point (say, $g(s,h)$) among $g(i,j)$ to G_0. If $\tilde{\mu}_h$ is a short part, we put the point $p(s,h)$ into $\tilde{S}_h$, assign other boundary elements (computed so far) suitably, and return. Otherwise, we replace J by $J \cup \{h\}$, make $p(s,h)$ as a boundary element, and continue.

Let us consider the time complexity of an insertion. In order to find the indices s and h, it suffices to find $\max_{i \in J, j \in K-J}\{p(i,j)_i - p(i,j)_j - (g_i - g_j)\}$. Thus, it takes $c|J| \cdot |K - J|$ arithmetic operations to find $g(s,h)$ (c is a constant). In the worst case, the total number of such operations is $c\sum_{i=1}^{k-1} i(k-i) = O(k^3)$ for an insertion. Besides, since at most k elements are re-assigned, $k(k-1)$ priority queues are updated. Hence, we have the following lemma:

Lemma 2. *The insertion of a point takes $O(k^3 + k^2 \log n)$ time.*

Remark. If we can make an assumption (although it is not always fair) that the probability that an index h is added to J is independent of the choice of $h \in K - J$, the amortized complexity of an insertion is reduced to $O(k^{2.5} + k^{1.5} \log n)$.

We obtain the following proposition immediately from Lemma 2:

Proposition 1. *The incremental algorithm finds a λ-splitter in $O(k^3 n + k^2 n \log n)$ time and $O(kn)$ space.*

Above algorithm outperforms the Fredman-Tarjan's algorithm [FT], if $k \leq n^{0.5}$.

Note 1. At an incremental step, we find a minimum weight path on a complete digraph K on k nodes. Each node a_i of K represent a region $T(G; i)$ of the current splitting, and the directed edge from a_i to a_j has the cost $c(i,j) = p(i,j)_i - p(i,j)_j - g_i + g_j$. This graph has no negative cycle because of Theorem 1. The famous Bellman-Ford algorithm [FF] finds a minimum weight path in $O(k^3)$ time. Currently, the best time complexity is $O(k^{2.38})$ applying the fast matrix computation [CW]. However, it still needs $O(k^3 + k^2 \log n)$ time to update g and K in the worst case. We can design an $O(k^{2.38} n + k^2 n \log n)$ time algorithm by another approach, which we omit in this version.

3 A naive randomized algorithm

In this section, we give a randomized algorithm with time bound $O(kn + k^{10/3} n^{2/3})$ based on the incremental algorithm in Section 2. Let us consider a set $\tilde{S}$ consisting of m randomly chosen points from S. We consider the splitting of $\tilde{S}$ such that the number of the points in the i-th region $\tilde{T}_i$ is $\lfloor \frac{m}{n}\lambda_i \rfloor$ or $\lfloor \frac{m}{n}\lambda_i \rfloor + 1$.

We would like to estimate the number n_i of points of S contained in $\tilde{T}_i$. For a real number a, we use the

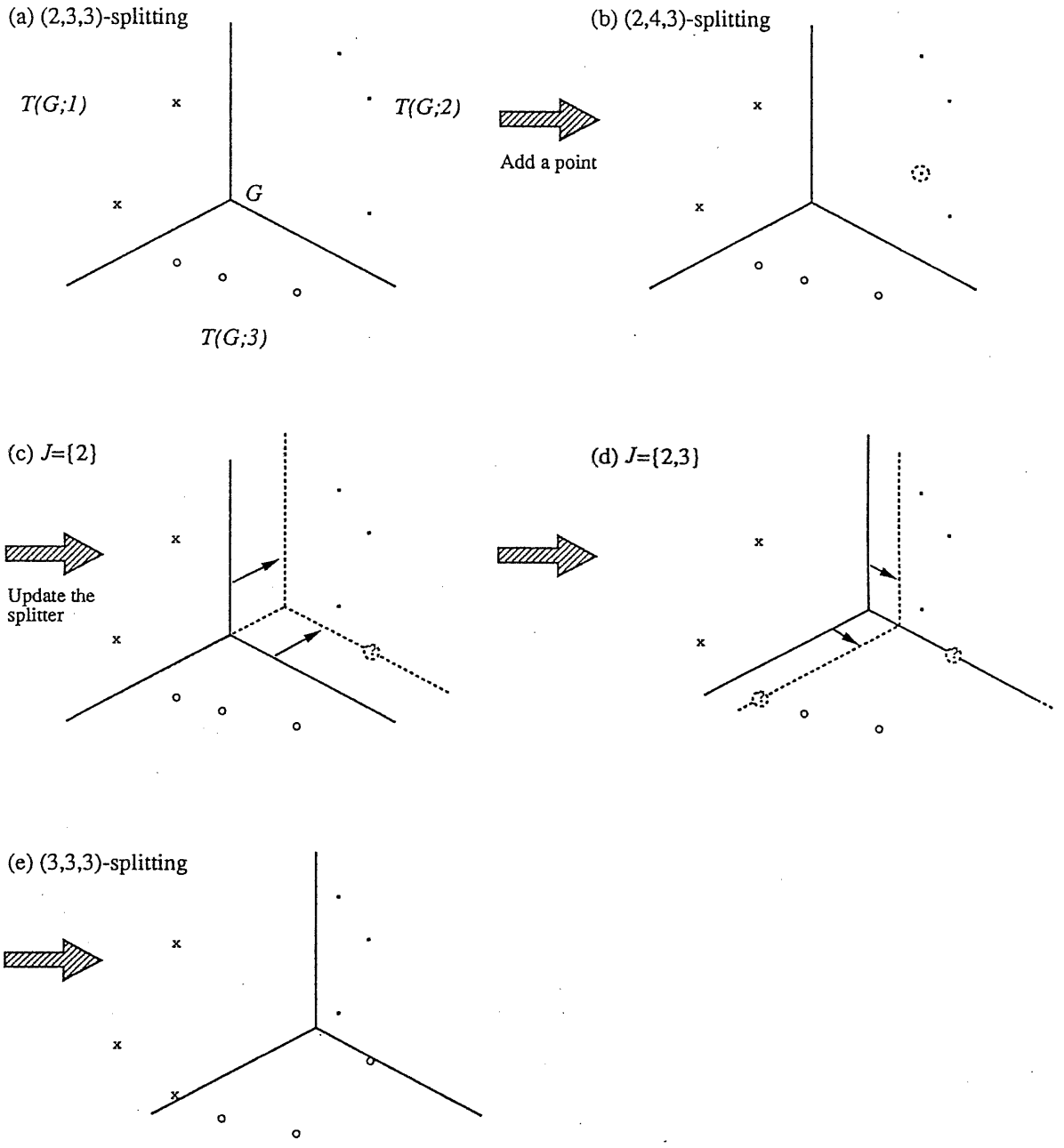

Figure 3: The insertion operation

notation $\|a\|$ to represent $\max(a, 0)$. The summation $\sum_{i=1}^{k} \|n_i - \lambda_i\|$ is called the excess.

Consider a sequence of m Bernoulli trials, where the success occurs with probability p. Let X be the random variable denoting the total number of success. The variable X follows the binomial distribution, and it is well known (see [CLR]) that

$$\Pr\Big(|X - E(X)| \geq r\Big) \leq e^{-r^2/4\sigma^2}. \qquad (3.1)$$

Here, σ is the variance, which is equal to $\sqrt{mp(1-p)}$.

Suppose we have a splitting of S such that its i-th region $\tilde{T}_i$ contains n_i points. Consider a random sampling of m points from S. Let X_i be the random variable denoting the number of sample points contained in $\tilde{T}_i$. The variable X_i almost follows the binomial distribution of success probability $p_i = n_i/n$. Although the distribution of X_i is not a true binomial distribution, (3.1) holds for it too (we omit the details). Thus, the probability that X_i is smaller than $p_i m - r_i$ is bounded by $e^{-r_i^2/4mp_i(1-p_i)}$.

Therefore, the expected number of $\|E(X_i) - X_i\|$ is bounded by

$$\begin{aligned} \int_0^m -x \frac{d}{dx} e^{-x^2/4mp_i(1-p_i)} dx &\leq \sqrt{\pi m p_i (1-p_i)} \\ &< \sqrt{\pi m n_i / n}. \qquad (3.2) \end{aligned}$$

Thus, the expected value of the difference between λ_i and the number n_i of points of S in $\tilde{T}_i$ is less than $\frac{n}{m} \times \sqrt{\pi m n_i / n}$, which is $\sqrt{\pi \lambda_i n/m} + o(\sqrt{\lambda_i n/m})$. The summation $\sum_{i=1}^{k} \sqrt{\pi \lambda_i n/m}$ attains its maximum $n\sqrt{\pi k/m}$ when $\lambda_i = n/k$ for all i. This number is larger than the expected value of the excess of the splitting. Thus, we obtain the following proposition.

Proposition 2. *The expected number of the excess is less than* $n\sqrt{\pi k/m}$.

On the basis of above observation, setting $m = k^{1/3}n^{2/3}$, we design the following algorithm:

Algorithm 1;
1: Randomly choose $k^{1/3}n^{2/3}$ points of S;
2: Find the splitter of these points;
3: Add all points to the splitting without updating the splitter;
4: Remove excess points (until excess = 0);
5: Insert the removed points, incrementally updating the splitter;
end

If we use an $O(\log n)$-time priority queue to maintain the order of elements in each region, the expected performance of this algorithm is $O(kn \log n + k^{10/3}n^{2/3} + k^{7/3}n^{2/3} \log n)$. If we use the *Fibonacci heap* [FT], which is a priority queue performing an insertion in $O(1)$ amortized time (other operations are done in $O(\log n)$ amortized time), the complexity can be improved to $O(kn + k^{10/3}n^{2/3} + k^{7/3}n^{2/3} \log n)$. Since the term $k^{7/3}n^{2/3} \log n$ never becomes the leading term in the above expression, we obtain the following:

Theorem 2. *The expected time complexity of Algorithm* 1 *is* $O(kn + k^{10/3}n^{2/3})$.

Note 2. If $k = 2$, the problem is equivalent to the selection problem of the λ_1-th largest element in a set of n real numbers. Therefore, Algorithm 1 can be regarded as a higher dimensional version of the "randomized selection algorithm", although its one-dimensional version ($k = 2$) is different from known such linear time algorithms [FR].

Note 3. The classical assignment problem, where $\lambda = (1, 1, \ldots, 1, n-k+1)$ can be solved in $O(nk+k^4)$ deterministic time by applying Algorithm 1, if we define the initial splitter such that every point is involved in the k-th region.

Let us consider the tail distribution analysis of Algorithm 1. From now on, we assume $\lambda_i = n/k$ for any $i = 1, 2, \ldots, k$. It is easy to see the general problem can be reduced to this special case by increasing k and n to at most $2k$ and $2n$ respectively.

Because of the assumption, $E(X_i) = m/k$. We would like to compute the probability

$$P(r) = \Pr\left(\sum_{i=1}^{k} \|E(X_i) - X_i\| \geq r\right).$$

The following is the key proposition:

Proposition 3. *If* $c \geq 32, P(\sqrt{ckm}) < e^{-ck/8}$.

Proof. We define the function $F(r) = e^{-kr^2/4m}$. Then, from (3.1),

$$\Pr\Big(E(X_i) - X_i \geq r\Big) \leq e^{-r^2/4\sigma^2} \leq e^{-kr^2/4m} = F(r).$$

Because of the unimodality of the binomial distribution function,

$$P(r) \leq \int_{y=r}^{\infty} \int_{\vec{r} \in Q_y} \prod_{i=1}^{k} -\frac{\partial F(r_i)}{\partial r_i} dV dy, \qquad (3.3)$$

where $\vec{r} = (r_1, r_2, \ldots, r_k), dV = dr_1 dr_2 \cdots dr_k$, and Q_y is the simplex defined by $0 \leq x_i \leq y$ for $i = 1, 2, \ldots, k$ and $x_1 + x_2 + \cdots + x_k = y$. We remark although $X_i (i = 1, 2, \ldots, k)$ are not independent of each other, (3.3) certainly holds. The right hand of (3.3) is

$$\int_{y=r}^{\infty} \int_{\vec{r} \in Q_y} \left(\prod_{i=1}^{k} \frac{kr_i}{2m} \right) e^{-k|\vec{r}|^2/4m} dV dy. \qquad (3.4)$$

Since both $\prod_{i=1}^{k} r_i$ and $-|\vec{r}|^2$ attain their maximums (on Q_y) if and only if $r_i = y/k$ for $i = 1, 2, \ldots, k$, we have

$$P(r) < \int_{y=r}^{\infty} V_y \left(\frac{y}{2m} \right)^k e^{-y^2/4m} dy, \qquad (3.5)$$

where $V_y = \sqrt{k} y^{k-1}/(k-1)!$ is the volume of Q_y.

If we set $r = \sqrt{ckm}$ and compute (3.5), we obtain

$$P(\sqrt{ckm}) < 2^{k-1}\sqrt{k} \left\{ e^{-ck/4} \sum_{j=0}^{k-1} \frac{1}{j!} \left(\frac{ck}{4} \right)^j \right\}. \qquad (3.6)$$

If $c \geq 4$, the right hand of (3.6) is smaller than

$$2^{k-1}\sqrt{k} \left\{ e^{-ck/4} \frac{c^{k-1} k^k}{(k-1)! 4^{k-1}} \right\}.$$

Applying Stirling's formula,

$$P(\sqrt{ckm}) < \frac{c^{k-1} k}{2^{k-1}\sqrt{2\pi}} e^{(1-c/4)k}. \qquad (3.7)$$

If $c > 32$, the right hand of (3.7) is smaller than $e^{-ck/8}$; hence we obtain Proposition 3. ∎

Corollary 1. *The number of updating of the splitter in Algorithm* 1 *is less than* $\sqrt{c} k^{1/3} n^{2/3}$ *with a probability larger than* $1 - e^{-ck/8}$ *if* $c \geq 32$.

4 More efficient implementation

From Proposition 3, we have the following lemma (proof is omitted):

Lemma 3. *Suppose that we have a splitting of* n^α *points. Let* c *be a constant larger than* 32. *If we add points until the excess becomes* $3\sqrt{ckn}\, n^{\beta-\alpha/2}$ *for* $\beta > \alpha$, *the number of added points is more than* n^β *with a probability larger than* $1 - 3e^{-ck/8}$.

On the basis of the above observation, we design the following algorithm ($c \geq 32$ is a constant):

Algorithm 2;
1: Choose $\sqrt{n}$ elements at random;
2: G = splitter of the above elements;
3: $t = \lfloor \log \log n \rfloor$;
4: $i = 1$;
repeat
5.1: **while** the excess is less than $2^{i-t}3\sqrt{ckn}$ **do**
5.1.1: Randomly add points;
end do
5.2: Remove the excess points;
5.3: Insert the removed points, updating the splitter G;
5.4: $i = i + 1$;
until no point remains;
end

Proposition 4. *The number of updating operations of the splitter in Algorithm* 2 *is less than* $15\sqrt{ckn}$ *with a probability larger than* $1 - 3e^{-ck/8} \log \log n$.

Proof. Initially, we have a splitting of $\sqrt{n}$ points. From Lemma 3, more than $2^{1-t} n^{3/4}$ points are added before the excess becomes $2^{1-t}3\sqrt{ckn}$ with a probability of at least $1 - 3e^{-ck/8}$. Similarly, suppose that $2^{-h} n^{1-2^{-j}}$ point has been already inserted; then, the algorithm adds more than $2^{(-h/2)-t+j+1} n^{1-2^{-j-1}}$ points before the excess becomes $2^{-t+j+1}3\sqrt{ckn}$ with a probability larger than $1 - 3e^{-ck/8}$. Thus, after the first t recursions, $M \geq 2^{-\xi} n^{1-2^{-t}}$ points, where $\xi = \sum_{i=1}^{t-1} 2^{-i} i$, are added. Since $t = \log \log n$ and $\xi < 2$, $M < n/(2 \times 2^2) = n/8$. The probability that all these t steps succeed is larger than $1 - 3te^{-ck/8}$.

The total excess in these t recursions is less than $6\sqrt{ckn}$. After we have found the splitter of $n/8$ points, the excess caused by the addition of the remaining points is less than $3\sqrt{8ckn}$ with a probability larger than $1-3e^{-ck/8}$. Proposition 4 has been proved. ∎

The probability $1-3e^{-ck/8}\log\log n$ is small if $n \gg k$. However, the complexity of the algorithm is not affected by the $\log\log n$ factor, as seen below:

Lemma 4. *The performance of Algorithm 2 is* $O(kn+\gamma^{0.5}k^{3.5}n^{0.5})$ *with probability* $1-3e^{-\gamma k}$, *where* γ *is any real number larger than* 4 *and less than* $n/(\log\log n)^7$.

Proof. If $k \geq \log\log n$, Lemma 4 immediately follows from Proposition 4. If $k \leq \log\log n$, replacing the constant c by $c' = \gamma\log\log n$ in the algorithm, the number of updates of splitters is less than $15\sqrt{\gamma kn}\log\log n$ with probability $1-3e^{-\gamma k}$. Since $kn > \gamma^{0.5}k^{3.5}n^{0.5}\log\log n$ if $k \leq \log\log n$, Lemma 4 holds for this case too. ∎

Theorem 3. *The performance of Algorithm 2 is* $O(kn+k^{3.5}n^{0.5})$ *with probability* $1-3e^{-cn^{0.2}}$ *for any constant* c. *The space complexity is* $O(kn)$.

Proof. If $k \geq n^{0.2}$, Theorem 3 is an immediate corollary of Lemma 4. If $k < n^{0.2}$, we set $\gamma = cn^{0.2}/k$ and apply Lemma 4 to obtain the theorem. ∎

5 Concluding remarks

We have developed a new method to find the minimum cost assignment on a complete bipartite graph between n nodes and k nodes. Our method is efficient if n is large enough compared with k. The time complexity shown here is the worst case one, and it is expected to be better in practice. We would like to apply the framework to several other optimization problems. Moreover, we have given a new type of the applications, which was presented as an open problem in [FT], of the Fibonacci heap. For our purpose, its *lazy deletion* version seems to be advantageous.

Acknowledgement

We wish to thank Kazumiti Numata, Akihisa Tamura, and Komei Fukuda for helpful discussions, which stimulated our interest on assignment problems. Kazuo Iwano of IBM Tokyo Research Lab. encouraged this work and gave helpful comments.

References

[CLR] Cormen, T., Leiserson, C., and Rivest, R., *Introduction to Algorithms,* The MIT Press, 1989.

[CW] Coppersmith, D. and Winograd, S., "Matrix multiplication via arithmetic progressions," *Proc. 9th ACM Symposium of Theory of Computing,* pp.1–6, 1987.

[FF] Ford, L. and Fulkerson, D., *Flows in Networks,* Princeton University Press, 1962.

[FR] Floyd, R. and Rivest, R., "Expected time bounds for selection," *Communications of ACM.*, 18, pp.165–172, 1975.

[FT] Fredman, M. and Tarjan, R., "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM.*, 34, pp.596–615, 1987.

[GT] Gabow, H. and Tarjan, R., "Faster scaling algorithms for network problems," *SIAM J. Comput.*, 18, pp.1013–1036, 1989.

[NT] Numata, K. and Tokuyama, T., "Splitting a configuration in a simplex," submitting, a preliminary version is in *Springer Lecture Notes in Comput. Sci.*, 450, pp.429–438, 1990.