Approximation Algorithms for Data Association Problem Arising from Multitarget Tracking

Naoyuki Kamiyama Tomomi Matsui

Department of Information and System Engineering, Faculty of Science and Engineering, Chuo University, Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan.
Email: {kamiyama, matsui}@iise.chuo-u.ac.jp

Abstract

In this paper, we discuss a data association problem arising from multitarget tracking. We formulate the problem as a multi-dimensional assignment problem and propose a polynomial time 1.8-approximation algorithm for simple case. We also propose a 3.7-approximation algorithm for general cases.

Keywords: multitarget tracking, data association problem, multi-dimensional assignment problem, approximation algorithm

1 Introduction

Multiple target tracking is a subject devoted to the estimation of the trajectory of targets. The main problem in multiple target tracking is a data association problem of determining which sensor measurements emanate from which target. In this paper, we discuss a data association problem arising from multiple target tracking.

At time $t = 1$ a sensor is turned on to observe the region. At each time instance $t \in \{1, 2, \ldots, k\}$, the sensor produces a report denoted by $V_t$. We assume that each report consists of measurements of $n$ targets. The actual type of measurement varies with the sensor. For example, a 2-dimensional radar measures range and azimuth of each potential target, a 3-dimensional radar that measures range, azimuth, and elevation, a 3-dimensional radar with Doppler measures these and the time derivative of range. We have a set of reports $\{V_1, V_2, \ldots, V_k\}$ such that each report consists of $n$ measurements of targets without a knowledge that which measurement emanates from which target. The problem then is to determine which measurements go with which targets.

The formulation of a data association problem requires the specification of edges $\{i, j\}$ defined by a pair of measurements $i$ and $j$. An edge $\{i, j\}$ represents an assignment between measurement $i$ from report $V_t$ and measurement $j$ from report $V_{t'}$ where $t \neq t'$. Each edge $e = \{i, j\}$ has a weight $w$, which is computed based on the dynamics of targets modeled from physical laws of motion. In this paper, we assume that edge weights satisfy triangle inequalities. For each target, a subset of measurements emanating from the target consists of $k$ measurements and meets every report $V_t$ $(t \in \{1, 2, \ldots, k\})$ in exactly one measurement. The data association problem is to find a partition of all the measurements such that each subset in the partition meets every report in exactly one measurement and which minimizes the sum of weights of edges connecting pairs of measurements in a mutual subset. The data association problem arising from target tracking appears in some papers (Poore 1994, Poore & Rijavec 1994, Poore & Gadaleta 1996, Storms & Spieksma 2003, Kuroki & Matsui 2009). In the next section, we give a mathematical formulation of our problem as a multi-dimensional assignment problem.

2 Multi-dimensional Assignment Problem

Let $\mathcal{F} = \{V_1, V_2, \ldots, V_k\}$ be a given set of reports. Given a positive integer $d$, we introduce a relation graph $\tilde{G}_d$ with a set of $k$ vertices $\mathcal{F} = \{V_1, V_2, \ldots, V_k\}$ and an edge set $\tilde{E}_d$ defined by

$$\tilde{E}_d = \{\{V_i, V_j\} \mid \exists k \leq i \neq j \leq n, \ j - i \leq d\}.$$  

Figure 1 shows relation graph $\tilde{G}_2$ with $k = 12$ vertices.

![Relation graph $\tilde{G}_2$ (k = 12).](image)

We assume that each report $V_i \in \mathcal{F}$ consists of $n$ vertices (measurements), i.e., $|V_i| = |V_j| = \cdots = |V_k| = n$. Now we introduce a graph for representing the set of measurements. A $k$-partite graph $G_d(V_1, V_2, \ldots, V_k; E)$ is obtained from relation graph $\tilde{G}_d$ by replacing

1. each vertex $V_i$ with a set of $n$ vertices, and
2. each edge with a complete bipartite graph $K_{n,n}$.

Figure 2 gives an example of $k$-partite graph where $d = 2, k = 12$ and $n = 3$.

More precisely, $G_d(V_1, V_2, \ldots, V_k; E)$ is defined by vertex sets $V_1, V_2, \ldots, V_k$, and an edge set

$$E_d = \bigcup_{\{V_i, V_j\} \in \tilde{E}_d} \{\{u, v\} \mid u \in V_i, v \in V_j\}.$$  

We introduce non-negative edge weights $w : E_d \to \mathbb{Z}_+$ satisfying triangle inequalities.

We denote the vertex set $V_1 \cup V_2 \cup \cdots \cup V_k$ of $G_d(V_1, V_2, \ldots, V_k; E)$ by $\tilde{V}$. For any vertex subset

---

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the 17th Computing: The Australasian Theory Symposium (CATS 2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 119, Alex Potanin and Taso Viglas, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.
In this section, we propose a simple heuristic algorithm. Given a spanning tree $T$ of the relation graph $G_d$, we define a heuristic algorithm $HT(T)$ as follows.

Algorithm $HA(T)$:

Step 1: For each edge $e = \{V_i, V_j\}$ in $T$, we find a minimum weight perfect matching $M(e)$ in the induced bipartite subgraph $G_d[V_i \cup V_j]$ of $G_d$.

Step 2: We construct a graph $(\hat{V}, \cup_{e \in T} M(e))$ and decompose the vertex set $\hat{V}$ into a family of connected components $\{Q_1, Q_2, \ldots, Q_n\}$.

Step 3: Output a feasible partition $\{Q_1, \ldots, Q_n\}$.

In the rest of this paper, $Q(T)$ denotes a feasible partition obtained by executing $HA(T)$. The sum of weights $w(Q_1) + \cdots + w(Q_n)$ is denoted by $w(Q(T))$.

We discuss the approximation ratio of $HA(T)$. First, we introduce an upper bound of $w(Q(T))$.

Since the edge weights of $G_d$ satisfy triangle inequalities, it is easy to show that $w(Q(T)) \leq \sum_{e \in E_d} w(M(e))a_T(e)$, and thus we have that:

$$w(Q(T)) \leq \sum_{e \in E_d} w(M(e))a_T(e) \leq \left( \sum_{e \in E_d} w(M(e)) \right) \left( \max_{e \in E_d} a_T(e) \right) \leq z^* \max_{e \in E_d} a_T(e).$$

From the above, the approximation ratio of $HA(T)$ is bounded by $\max_{e \in E_d} a_T(e)$. \hfill \Box

4 Approximation Algorithm for $\tilde{G}_2$

In this section, we discuss a case that $d = 2$.

First, we give a spanning tree such that the heuristic algorithm described in the previous section becomes a 3-approximation algorithm. Let $E^1 = \{V_1, V_2, V_3, \ldots, V_{k-1}, V_k\}$ and $E^0 = \tilde{E}_2 \setminus E^1$. Spanning tree $T_s$: Let $T_s$ be a spanning tree (Hamilton path) on $G_2$ induced by $E^1$.

Figure 3 shows an example of $T_s$ and a graph with parallel edges whose multiplicities are defined by (1).

Then we have the following.

Corollary 4.1 An approximation ratio of Algorithm $HA(T_s)$ is less than or equal to 3.
Proof: It is easy to see that
\[ a_{T_*}(e) \leq \begin{cases} 
3, & \text{if } e \in T_* = \hat{E}^1, \\
0, & \text{if } e \in \hat{E}_d \setminus T_* = \hat{E}_d^0.
\end{cases} \]
and so \( \max_{e \in \hat{E}_d} a_{T_*}(e) \leq 3. \)

Next, we introduce three additional spanning trees on \( \hat{G}_2 \) appearing in Figure 4. In the rest of this section, we assume that \( k \) is a multiple of \( 6 \). We can drop this assumption easily.

Spanning tree \( T_i \ (i = 0, 1, 2) \): First, we introduce an infinite set
\[
P_i = \bigcup_{j \in \mathbb{Z}} \{ \{V_{2j+1}, V_{3j+1}, V_{3j+2}, V_{3j+3}, V_{3j+4}, V_{3j+5}, V_{3j+6}\}, \{V_{4j+2}, V_{5j+2}, V_{5j+3}, V_{5j+4}, V_{5j+5}, V_{5j+6}\}, \{V_{6j+1}, V_{6j+2}, V_{6j+3}, V_{6j+4}, V_{6j+5}, V_{6j+6}\} \}
\]
which forms a path in a graph with an infinite vertex set \( \{V_i \mid i \in \mathbb{Z}\} \). We set \( \hat{P}_i = P_i \cap \hat{E}_2 \). We define a spanning tree \( T_i \ (i = 0, 1, 2) \) by
\[
T_i = \begin{cases} 
\hat{P}_i, & \text{if } i = 0, 1, \\
\hat{P}_i \cup \{\{V_1, V_2\}, \{V_{k-1}, V_k\}\}, & \text{if } i = 2.
\end{cases}
\]

Figure 4 shows examples of three spanning trees and graphs with parallel edges whose multiplicities are defined by (1).

It is easy to see that
\[
a_{T_i}(e) \leq \begin{cases} 
0, & \text{if } e \in \hat{E}_2 \setminus T_i, \\
3, & \text{if } e \in T_i \cap \hat{E}_d^0, \\
5, & \text{if } e \in (T_i \cap \hat{E}_d^1) \setminus \{\{V_1, V_2\}, \{V_{k-1}, V_k\}\}, \\
3, & \text{if } (e, i) = (\{V_1, V_2\}, 0) \\
\text{or } (e, i) = (\{V_{k-1}, V_k\}, 1), \\
0, & \text{if } (e, i) = (\{V_1, V_2\}, 1) \\
\text{or } (\{V_{k-1}, V_k\}, 0), \\
2, & \text{if } e \in \{\{V_1, V_2\}, \{V_{k-1}, V_k\}\} \\
\text{and } i = 2.
\end{cases}
\]

Now, we describe our algorithm.

**Algorithm A1:**

**Step 1:** For each spanning tree \( T \in \{T_0, T_1, T_2\} \), we execute heuristic algorithm HA(T) and obtain four feasible partitions \( Q(T_0), Q(T_1), Q(T_2) \) and \( Q(T_3) \).

**Step 2:** We output a feasible partition, denoted by \( Q^* \), in \( \{Q(T_0), Q(T_1), Q(T_2)\} \) which attains the value
\[
\min\{w(Q(T_*)), w(Q(T_0)), w(Q(T_1)), w(Q(T_2))\}
\]

It is easy to see that the computational effort of our algorithm is bounded by a polynomial of the problem input size.

In the rest of this section, we estimate the approximation ratio of Algorithm A1. Recall that for each edge \( e = \{V_i, V_j\} \in \hat{E}_2 \), \( w(M(e)) \) denotes the optimal value of minimum weight perfect matching problem defined on the bipartite induced subgraph \( G[V_i \cup V_j] \).

Thus, it is obvious that \( \sum_{e \in \hat{E}_2} w(M(e)) \) gives a lower bound of the optimal value of a given MDA problem. The feasible partition \( Q^* \) obtained by Algorithm A1 satisfies that
\[
w(Q^*) = \min\{w(Q(T_*)), w(Q(T_0)), w(Q(T_1)), w(Q(T_2))\}
\leq \left( \frac{1}{10} \right) w(Q(T_*)) + \left( \frac{3}{10} \right) w(Q(T_0)) + \left( \frac{3}{10} \right) w(Q(T_1)) + \left( \frac{3}{10} \right) w(Q(T_2))
\]
\[
\leq \sum_{e \in \hat{E}_2} w(M(e)) \left( \frac{1}{10} a_{T_*}(e) + \frac{3}{10} a_{T_0}(e) + \frac{3}{10} a_{T_1}(e) + \frac{3}{10} a_{T_2}(e) \right).
\]
It is obvious that for each edge $e \in \tilde{E}^{0}$,
\[
\left(\frac{1}{10}\right) a_{T_{e}}(e) + \left(\frac{3}{10}\right) (a_{T_{e}}(e) + a_{T_{e}}(e) + a_{T_{e}}(e)) \leq \frac{3}{10} + \frac{3(3 + 3 + 0)}{10} = 18/10.
\]
When $e$ is an edge in $\tilde{E}^{1} \setminus \{(V_{1}, V_{2}), \{V_{k-1}, V_{k}\}\}$, we can show that
\[
\left(\frac{1}{10}\right) a_{T_{e}}(e) + \left(\frac{3}{10}\right) (a_{T_{e}}(e) + a_{T_{e}}(e) + a_{T_{e}}(e)) \leq \frac{3}{10} + \frac{3(5 + 0 + 0)}{10} = 18/10.
\]
If $e$ is either $\{V_{1}, V_{2}\}$ or $\{V_{k-1}, V_{k}\}$, we have that
\[
\left(\frac{1}{10}\right) a_{T_{e}}(e) + \left(\frac{3}{10}\right) (a_{T_{e}}(e) + a_{T_{e}}(e) + a_{T_{e}}(e)) \leq \frac{3}{10} + \frac{3(2 + 3 + 0)}{10} = 18/10.
\]
The above inequalities imply that
\[
w(Q^{*}) \leq \sum_{e \in E_{d}} w(M(e)) \left(\frac{18}{10}\right) \sum_{e \in E_{d}} w(M(e)) \leq 1.8z^{*},
\]
where $z^{*}$ denotes the optimal value of a given MDA problem. Thus, we have shown the following.

**Theorem 4.2** Algorithm A1 is a polynomial time 1.8-approximation algorithm.

**5 Lower Bound of Approximation Ratio**

In this section, we discuss a theoretical advantage of Algorithm A1.

Algorithm A1 is a 1.8-approximation algorithm which uses four spanning trees. It is natural to ask an existence of a set of spanning trees which gives a similar approximation algorithm with a better approximation ratio. We introduce an algorithmic artificial algorithm with a given set of spanning trees $T^{*}$ on $G_{d}$.

**Algorithm A(T):**

For each spanning tree $T \in T'$, we execute heuristic algorithm HA(T) and obtain a feasible partition $Q(T)$. We output a feasible partition which attains the value $\min_{T' \in T} w(Q(T))$.

Clearly, if we use the set of all the spanning trees, denoted by $T$, in the relation graph $G_{d}$, the best possible approximation ratio is obtained. Unfortunately, the number of spanning trees in $T$ grows exponentially with respect to $k$, and thus Algorithm A(T) is an exponential time algorithm. In the following, we estimate an approximation ratio of Algorithm A(T) in a similar way with the proof of Theorem 4.2.

Let $A$ be a matrix whose rows are index by $E_{d}$, columns are indexed by $T$, and satisfies that a column vector index by $T \in T$ is $a_{T}$, which denotes the edge-multiplicities defined by (1). We introduce a non-negative weight vector $x \in \mathbb{R}^{T}$ satisfying that the sum total is equal to 1 (i.e., $x \geq 0$ and $1^{\top} x = 1$).

Then a solution obtained by Algorithm(T) satisfies that
\[
\min_{T \in T} w(Q(T)) \leq \sum_{T \in T} x_{T} w(Q(T)) \leq \sum_{T \in T} x_{T} \sum_{e \in E_{d}} w(M(e)) a_{T}(e) = \sum_{e \in E_{d}} w(M(e)) \left(\sum_{T \in T} x_{T} a_{T}(e)\right) \leq \sum_{e \in E_{d}} w(M(e)) \left(\max_{e' \in E_{d}} \sum_{T \in T} x_{T} a_{T}(e')\right) = \sum_{e \in E_{d}} w(M(e)) \left(\max_{e' \in E_{d}} \sum_{T \in T} x_{T} a_{T}(e')\right) = z^{*} \left(\max_{e \in E_{d}} \sum_{T \in T} x_{T} a_{T}(e')\right) = z^{*} \max_{e \in E_{d}} (Ax)_{e},
\]
where $z^{*}$ denotes the optimal value and $(Ax)_{e}$ denotes an element of vector $Ax$ indexed by $e \in E_{d}$. Thus, the approximation ratio of $A(T)$ is bounded by the maximum of elements in the vector $Ax$. Conversely, if we have a non-negative vector $x' \in \mathbb{R}^{T}$ satisfying $1^{\top} x' = 1$, we can construct Algorithm A(T) by setting $T' = \{T' \in T \mid x'(T') > 0\}$ whose approximation ratio is bounded by the maximum of elements in $Ax'$. For example, when $d = 2$, Theorem 4.2 showed that a weight vector defined by
\[
x'(T') = \begin{cases} 1/10, & \text{if } T' = T_{1}, \\ 3/10, & \text{if } T' \in \{T_{0}, T_{1}, T_{2}\}, \\ 0, & \text{otherwise}, \end{cases}
\]
satisfies that the maximum of elements in $Ax'$ is bounded by 1.8 and thus Algorithm A(T) becomes a 1.8-approximation algorithm.

We can find a best weight vector $x$ for estimating the approximation ratio of Algorithm A(T) by solving the following linear programming problem
\[
P(k): \text{minimize } \eta \ \text{subject to} \ Ax - 1 \eta \leq 0, \\
1^{\top} x = 1, \\
x \geq 0,
\]
where $k$ is the number of vertices in the relation graph $G_{d}$. If we denote the optimal value of Problem $P(k)$ by $\eta^{*}(k)$, the approximation ratio of Algorithm A(T) is less than or equal to $\eta^{*}(k)$. Here we note that Problem $P(k)$ is equivalent to a standard formulation of a problem to find a Nash equilibrium of 2 persons zero sum game.

In the rest of this section, we assume that $d = 2$ and show the following theorem.

**Theorem 5.1** When $d = 2$, the optimal value $\eta^{*}(k)$ of Problem P(k) satisfies that
\[
(1.8 \frac{k - 3}{k - 1}) \leq \eta^{*}(k) \leq 1.8 \text{ and } \lim_{k \to \infty} \eta^{*}(k) = 1.8.
\]

The above theorem implies that Algorithm A1 attains the best possible approximation ratio asymptotically.
and so the set of four spanning trees employed in Algorithm A1 is asymptotically best.

A dual problem of $P(k)$ is defined by

$$D(k): \text{maximize } \xi$$

subject to

$$y^\top A - 1\xi \geq 0,$$

$$y^\top 1 = 1,$$

$$y \geq 0.$$  

The weak duality theorem says that any dual feasible solution $(y, \xi)$ of $D(k)$ satisfies $\xi \leq \eta^*(k)$. We introduce a dual feasible solution which gives an asymptotically tight lower bound of $\eta^*(k)$.

**Lemma 5.2** When $d=2$, a dual solution $(y', \xi')$ defined by

$$y'(e) = \begin{cases} 
0.6 & \text{if } e \in \tilde{E}^1, \\
0.4 & \text{if } e \in \tilde{E}^0, \\
0.2 & \text{if } e \in \tilde{E}^{-1},
\end{cases}$$

and $\xi' = (1.8)k - 3$ is feasible to Problem $D(k)$.

Proof: Clearly, $y'$ is non-negative and sum total is equal to 1. Thus, we only need to show that

$$\forall T \in T, y^\top a_T \geq \xi'.$$

We denote the number of edges in $T \cap \tilde{E}^0$ by $\zeta_T$. It is obvious that

$$\forall e \in \tilde{E}^0 \setminus \{\{V_1, V_3\}, \{V_k-2, V_k\}, e \in T \rightarrow |c(T, e)| \geq 3.$$  

Thus, we have

$$\sum_{e \in \tilde{E}^0} y'(e)a_T(e) = \frac{0.4}{k-2} \sum_{e \in \tilde{E}^0 \cap T} |c(T, e)| \geq \frac{0.4}{k-2}(\zeta_T - 3) = (1.2)(\zeta_T - 2) + (0.6)\frac{3k - 2\zeta_T - 5}{k-1}.$$  

Since every elementary cycle in $\tilde{G}_2$ includes exactly two edges in $\tilde{E}^1$, it is easy to show that

**C1:** $\forall e_1 \in \tilde{E}^1 \setminus T, \exists f \in \tilde{E}^1 \cap T, e_1 \in c(T, f)$.

**C2:** $\forall e_2 \in \tilde{E}^0 \setminus T, \exists f' \in \tilde{E}^1 \cap T, e_2 \in c(T, f'), f' \in \tilde{E}^1 \setminus T, e_2 \in c(T, f')$ and $f' \neq f''$.

Figure 5 shows examples of above properties.

**Figure 5:** Examples of Properties C1 and C2.

The pair of properties implies that

$$\sum_{e \in \tilde{E}^1} y'(e)a_T(e) = \frac{0.6}{k-1} \sum_{e \in \tilde{E}^1 \cap T} |c(T, e)| \geq \frac{0.6}{k-1}(|\tilde{E}^1| + 2|\tilde{E}^0 \setminus T|) = \frac{0.6}{k-1}(k - 1 + 2(k - 2 - \zeta_T)) = (0.6)\frac{3k - 2\zeta_T - 5}{k-1}.$$  

From the above, we have that

$$y^\top a_T = \sum_{e \in \tilde{E}^0} y'(e)a_T(e) + \sum_{e \in \tilde{E}^1} y'(e)a_T(e) \geq (1.2)\frac{\zeta_T - 2}{k-2} + (0.6)\frac{3k - 2\zeta_T - 5}{k-1} \geq (1.2)(\zeta_T - 2) + (0.6)(3k - 2\zeta_T - 5) = (1.8)\frac{k-3}{k-1} = \xi'.$$

Thus, the solution $(y', \xi')$ is dual feasible.

Now we show Theorem 5.1

**Proof of Theorem 5.1:** Theorem 4.2 shows that a vector $(x', \eta')$ defined by

$$x'(T') = \begin{cases} 
1/10, & \text{if } T' = T, \\
3/10, & \text{if } T' \notin \{T_0, T_1, T_2\}, \text{ and } \eta' = 1.8 \\
0, & \text{otherwise},
\end{cases}$$

is feasible to $P(k)$ and so $\eta^*(k) \leq 1.8$.

Lemma 5.2 and the weak duality theorem imply that $\eta^*(k) \geq \xi' = (1.8)\frac{k-3}{k-1}$ and thus

$$1.8 \geq \lim_{k \to \infty} \eta^*(k) \geq \lim_{k \to \infty} (1.8)\frac{k-3}{k-1} = 1.8.$$  

From the above, we have a desired result.

**6 Algorithm for General Case**

In this section, we propose an approximation algorithm for general case that $d \geq 3$. In the rest of this paper, we assume that $k \geq 2d + 1$. We introduce a set of pairs of integers

$$D = \{1, 2, \ldots, d+1\} \times \{1, 2, \ldots, d\}.$$  

In our algorithm, we generate $(d+1)d$ spanning trees $\{T(r, s) \mid (r, s) \in D\}$ of $\tilde{G}_d$ and execute the heuristic algorithm proposed in Section 3. Figure 6 shows 12 spanning trees to be generated when $d = 3$ and $\tilde{G}_d$ has 10 vertices. A precise definition of required spanning trees appears in the following algorithm.

**Algorithm B:**

**Step 1:** For each pair $(r, s) \in D$, we execute the following. We define a set of vertices

$$\mathcal{F}_r = \{V_i \in \mathcal{F} \mid i = r \mod (d+1),$$

**Figure 6:** Required spanning trees when $d = 3$. (Black vertices denote anchor vertices.)
that in a tree.

Using this weight vector, we have that

\[ \sum_{i,j} w(Q,r,s) \leq 1 \]

It is obvious that the vector \( \mathbf{x} \) satisfies that the sum total is equal to 1 and \( x_1 > x_2 > \cdots > x_{d+1} \). By using this weight vector, we have that

\[ w(Q) = \min_{w(Q,r,s)} \sum_{(r,s) \in D} \frac{x_r}{d} w(Q(r,s)) \]

\[ \leq \sum_{(r,s) \in D} \left( \frac{x_r}{d} \sum_{e \in E_d} w(M(e)) a_{T(r,s)}(e) \right) \]

\[ = \sum_{e \in E_d} \left( w(M(e)) \sum_{(r,s) \in D} \frac{x_r}{d} a_{T(r,s)}(e) \right). \]

Case 1: Consider the case that \( e = \{V_i, V_j\} \in \tilde{E}_d \) satisfies \( i < j \leq d + 1 \). The edge \( \{V_i, V_j\} \) is contained in a tree \( T(r,s) \) if and only if \( r \in \{i, j\} \). We can show that

\[ a_{T(r,s)}(e) \leq \begin{cases} 
\frac{d-1+i}{d}, & \text{if } r = j, \\
\frac{d(d+1)}{2}, & \text{if } r = i \\
2d, & \text{and } i + d + 1 - s = j, \\
0, & \text{otherwise}.
\end{cases} \]

It implies that

\[ \sum_{(r,s) \in D} \frac{x_r}{d} a_{T(r,s)}(e) \leq \frac{x_j}{d} \left( d(d+1) \right) + \frac{x_i}{d} \left( \frac{d+1}{2} + 2d - 2 \right) = x_i \left( \frac{7}{2}d - (5/2) + i \right) \leq \frac{1}{\theta} \]

Case 2: Consider the case that \( e = \{V_i, V_j\} \in \tilde{E}_d \) satisfies \( i < j \) and \( d + 1 < j \). A tree \( T(r,s) \) includes \( e = \{V_i, V_j\} \) if only if either \( V_i \) or \( V_j \) is an anchor vertex (a vertex contained in the set \( F_r \)). We can show that

\[ a_{T(r,s)}(e) \leq \begin{cases} 
\frac{d(d+1)}{2}, & \text{if } V_j \in F_r \\
\frac{d(d+1)}{2}, & \text{and } i = j - s, \\
2d, & \text{if } V_i \in F_r \\
0, & \text{and } i + d + 1 - s = j, \\
\end{cases} \]

From the above, we have that

\[ \sum_{(r,s) \in D} \frac{x_r}{d} a_{T(r,s)}(e) \leq \frac{x_j}{d} \frac{d(d+1)}{2} + \frac{x_i}{d} \frac{d(d+1)}{2} + \frac{x_i}{d} (d-1)2d \]

\[ = x_i \left( \frac{7}{2}d - (5/2) + i \right) \leq \frac{1}{\theta} \]

Thus, an approximation ratio of Algorithm B is bounded by \( \frac{13d-5}{14d-6} / \theta \). Table 1 shows our upper bound of approximation ratio when \( d \in \{3, 4, 5\} \).

<table>
<thead>
<tr>
<th>( d )</th>
<th>( 1/\theta )</th>
<th>upper bound of ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2.59502</td>
<td>2.45086</td>
</tr>
<tr>
<td>4</td>
<td>2.87223</td>
<td>2.69990</td>
</tr>
<tr>
<td>5</td>
<td>3.05690</td>
<td>2.86584</td>
</tr>
<tr>
<td>( \infty )</td>
<td>( &lt; 3.97908 )</td>
<td>( &lt; 3.69486 )</td>
</tr>
</tbody>
</table>

Lastly, we discuss general case that \( d \geq 3 \). The upper bound of an approximation ratio satisfies that

\[ \frac{13d-5}{14d-6} / \theta = \frac{1}{\theta} \frac{d+1}{\sum_{r=1}^{(13/2)d-(5/2)+d} \frac{1}{r}} \]

\[ \leq \frac{13d-5}{14d-6} / \left( \frac{\int (9/2)d-(3/2) dz}{z} \right) \]

\[ = \frac{13d-5}{14d-6} / \left( \frac{\ln (9d-3)}{z} \right). \]
Denote \( \left( \frac{13d-5}{14d-6} \right) / \left( \ln \frac{9d-3}{7d-3} \right) \) by \( h(d) \). It is easy to show that \( h(d) \) is a non-decreasing function and thus

\[
h(d) \leq \lim_{d \to \infty} h(d) = \left( \frac{13}{14} \right) / \left( \ln \frac{9}{7} \right) \leq 3.69486.
\]

From the above discussion, we have the following.

**Theorem 6.1** If the number of vertices of relation graph \( \tilde{G}_d \) is greater than or equal to \( 2d + 1 \), the approximation ratio of Algorithm B is bounded by \( \left( \frac{13d-5}{14d-6} \right) / \theta \leq 3.69486 \) where

\[
\theta = \sum_{r=1}^{d+1} \frac{1}{(7/2)d - (5/2) + r}.
\]

7 Conclusion

In this paper, we consider a data association problem arising from multi-target tracking. We formulated the problem as a multidimensional assignment problem defined on a multipartite graph \( G_d \). We described a simple heuristic algorithm using a spanning tree in the relation graph \( \tilde{G}_d \).

When \( d = 2 \), we construct a specified set of four spanning trees which gives a 1.8-approximation algorithm. We also show that the proposed set of spanning trees is asymptotically best.

For a general case that \( d \geq 3 \), we proposed an approximation algorithm whose approximation ratio is bounded by 3.7.

Acknowledgement

We thank Yoshitaka Sugiura for helpful discussions.

References


