Track-to-Track Association and Ambiguity Management in the Presence of Sensor Bias

Dimitri J. Papageorgiou
The Raytheon Company
Integrated Defense Systems
Woburn, MA, U.S.A.
Dimitri_J_Papageorgiou@raytheon.com

Michael Holender
The Raytheon Company
Integrated Defense Systems
Woburn, MA, U.S.A.
Michael_N_Holender@raytheon.com

Abstract – The track-to-track association problem is to determine the pairing of sensor-level tracks that correspond to the same true target from which the sensor-level tracks originated. This problem is crucial for multisensor data fusion and is complicated by the presence of individual sensor biases, random errors, false tracks, and missed tracks. A popular approach to performing track-to-track association between two sensor systems is to jointly optimize the a posteriori relative bias estimate between the sensors and the likelihood of track-to-track association. Algorithms that solve this problem typically generate the $K$ best bias-association hypotheses and corresponding bias-association likelihoods. In this paper, we extend the above approach in two ways. First, we derive a closed-form expression for computing “pure” track-to-track association likelihoods, as opposed to bias-association likelihoods which are weighted by a unique relative bias estimate. Second, we present an alternative formulation of the track-to-track association problem in which we optimize solely with respect to association likelihoods. These results facilitate what is commonly known as system-level track ambiguity management.

Keywords: Track-to-track association, data fusion, 0-1 nonlinear optimization, semidefinite programming.

1 Introduction

A fundamental problem in multisensor data fusion is associating data from different sensor systems in the presence of sensor bias, random errors, false alarms, and missed detections. Level 1 of the JDL Fusion [4] model encompasses many different processes while attempting the overall goal of Object Refinement. Raw data is input into the system and the first problem that is considered is determining what that data refers to and which elements of data associate to one another. In our case, associating data is crucial to providing a coherent, integrated picture to the user, as well as pertinent track and attribute information about various targets of interest.

In the missile defense domain, a typical multiple target tracking (MTT) system is compromised of a suite of heterogeneous sensor systems and one or more fusion nodes which receive and process the raw data provided to the system. Each sensor processes its own data to generate and maintain sensor-level (also known as locally-fused) tracks, while each fusion node fuses the sensor-level tracks into a set of system-level tracks. This architecture is hierarchical in nature and has become a widely accepted system design choice in many circles because of the lack of single points of failure and information processing bottlenecks [11].

It is within the context of this hierarchical MTT system that the track-to-track association problem becomes so crucial. Before continuing forward with the fusion process, a fusion node must first have high confidence that the track data (from different sensors) to be combined correspond to the same target. A host of factors can complicate this association. First, participating sensors operating under different phenomenology may track different subsets of the truth. Second, closely-spaced objects may be difficult to resolve due to individual sensor sensitivity. A third issue concerns error due to sensor biases, which result from misalignment of measurement axes and sensor location error often arise and are difficult to estimate. Proper estimation and removal of this error is important to making correct assignments [2, 9, 10].

The track-to-track association problem is to determine the pairing of sensor-level tracks that correspond to the same true targets from which sensor-level tracks originated. The focus of this paper concerns track-to-track association between two sensor systems in the larger context of system-level tracking and discrimination. Figure 1 depicts a simple example of the track-to-track association problem in which it is imperative to account for the presence of sensor bias. Sensor $A$ sees four tracks, whereas sensor $B$ sees seven, including the four seen by sensor $A$. The positional error volume of each sensor’s tracks can be viewed as an ellipsoid in 3D.
Despite the fact that several ellipsoids overlap, the true relative bias is such that no two corresponding track volumes overlap in the graphic.

Figure 1. The track-to-track association problem using positional densities only.

Our point of departure is a track-to-track association problem posed by Levedahl [7], which he termed the Global Nearest Pattern Matching (GNPM) problem. The primary purpose of the GNPM problem was for a one-time sensor-to-sensor handover in which one sensor system transmits a frame of data to another sensor upon which a correlation algorithm is employed to correlate the information with its local track database. The novelty of Levedahl’s approach, as well as Kenefic’s [6] before him, was the explicit incorporation of a relative bias term into the likelihood function used to perform track-to-track association. For years, researchers and practitioners had acknowledged the existence of systematic sensor biases, but the vast majority of algorithms developed to resolve the problem were sequential in nature. That is, correlation algorithms would first attempt to estimate and remove the relative sensor bias, and then perform track-to-track association by solving a standard two-dimensional linear assignment problem (e.g., [17]). The fundamental drawback of this sequential approach lies in the infamous “chicken or the egg dilemma”: The determination of the relative bias is inextricably linked with the correct track-to-track association and vice versa. Kenefic and Levedahl formally coupled the two problems and the resulting formulation has seen increasing use within the MTT community.

In the missile defense domain, system-level tracking carries with it inherent uncertainties as heterogeneous sensors with different viewing geometries observe closely-spaced objects disperse in a random manner. This inherent uncertainty in track formation and association has led to an area known as track ambiguity management. Since identifying a single bias-association hypothesis whose likelihood “stands out” from the other hypotheses is an elusive task given this uncertainty, a secondary (and arguably less optimistic / more pragmatic) objective is to enumerate a subset of \( K \) bias-association hypotheses so that individual track pairings can be evaluated. Specifically, if there are a number of highly likely hypotheses whose likelihoods differ by negligible amounts, it is beneficial from a system-level perspective to know if there is a subset of track pairings that are common among all or most of the \( K \) best hypotheses. For example, in missile defense, one is often most interested in correctly associating a subset of the tracks, namely those that may represent objects of concern. Going a step further, it may be beneficial at the system-level for a correlation algorithm to quantify the likelihood that sensor \( A \) track \( i \) should be paired with sensor \( B \) track \( j \). Unfortunately, one cannot produce these likelihoods using the bias-association likelihoods produced by an algorithm for the GNPM problem as will be discussed in Section 3.

In this work, we make progress toward track-to-track fusion by addressing the multisensor association problem. The outline and the main contributions of this paper are as follows: First, after describing the salient features of our track-to-track association problem in Section 2, we show how to compute a “pure” track-to-track association likelihood in Section 3 when the relative bias between sensors is assumed to have a Gaussian prior. In Sections 4 and 5, we present an alternative track-to-track association problem formulation and argue that it is important for system-level track ambiguity management in missile defense. In Section 6, we propose a branch-and-bound approach for solving our track-to-track association problem to optimality using semidefinite programming relaxations at each node to provide increasingly better lower bounds. We believe this last contribution is important since semidefinite programming methods have not received much exposure in the multisensor data fusion literature.

2 Problem Description

Consider two independent sensor systems – sensor \( A \) and sensor \( B \) – tracking an unknown number of targets in space. Let \( N_A \) and \( N_B \) denote the set of tracks formed by sensor \( A \) and \( B \), respectively. Without loss of generality, we assume throughout that \( n_A = |N_A|, n_B = |N_B|, \) and \( n_A \leq n_B \). Due to a host of factors, including geometry and sensor resolution, the number of tracks formed by sensor \( A \) and \( B \) will often differ from the true number of targets and from one another. Let \( \mathbf{x}^i_A \) and \( \mathbf{P}_i \), for \( i \in N_A \), denote the state estimate and error covariance matrix, respectively, of the \( i \)th sensor \( A \) track. Similarly, let \( \mathbf{x}^j_B \) and \( \mathbf{Q}_j \), for \( j \in N_B \), denote the state estimate and covariance matrix, respectively, of the \( j \)th sensor \( B \) track. We assume all state estimates and covariance matrices have been extrapolated to a common time point and have been converted to a
A key assumption in this track-to-track association framework is that each track state is corrupted by a constant, but unknown, sensor bias \([3, 7, 11, 14]\). Ideally, these individual sensor biases would be estimated and removed prior to performing track-to-track association, but this is not always possible \([2, 9]\). Consequently, a distinguishing facet of this approach is our attempt to estimate the inter-sensor bias, or the relative bias, between the two sensors via maximum a posteriori (MAP) estimation, i.e., this is a Bayesian estimation framework. The relative bias \(b\) is modeled as a Gaussian random vector having mean \(0\) and covariance \(R\).

We denote a track-to-track association by the vector \(j\). Consequently, the association of the \(i\)th track in \(N_A\) with the \(j\)th track in \(N_B\) is denoted by \((i, j)\). It is convenient to think of the pair \((i, j)\) as an undirected arc in a bipartite graph and the vector \(j\) as a compact notation for writing \(\{(1, j_1), (2, j_2), \ldots, (n_A, j_{n_A})\}\). It is possible that the \(i\)th track in \(N_A\) is not assigned to any track in \(N_B\), in which case we still write \((i, j_i)\), but \(j_i = 0\). We refer to such an assignment as a null assignment, or by saying that track \(i\) was assigned to the dummy track. It is implicitly assumed that at most one sensor \(A\) track can be assigned to a sensor \(B\) track and vice versa. We refer to the pair \((b, j)\) as a bias-association hypothesis, a hypothesis, or a solution to the GNPM problem.

A popular objective for track-to-track association is to simultaneously find the most likely track-to-track association and relative bias estimate. To do so, a likelihood function is needed to compare different solutions. As derived in \([7]\), the likelihood function for the GNPM problem is based upon the marriage of an a posteriori bias estimation problem and the standard two sensor track-to-track association problem. The first term

\[
 -\frac{e^{-b^TR^{-1}b/2}}{(2\pi)^{D/2}/\sqrt{|R|}}
\]

(where \(|R|\) denotes the determinant of \(R\)) is nothing more than a prior probability density on the relative bias. The second term consists of the sum of the incremental likelihoods of track assignment. Specifically, given a bias estimate \(b\), the likelihood of assigning track \(i \in N_A\) and track \(j \in N_B\) is

\[
 \beta_T P_{AB}^k \frac{e^{-d_{ij}^2(b)/2}}{(2\pi)^{D/2}/\sqrt{|S_{ij}|}}
\]

where \(\beta_T\) is the target density, i.e., the number of targets per unit volume in \(D\)-dimensional space; \(P_{AB}\) is the probability that a target is tracked by sensor \(A\) and sensor \(B\); \(S_{ij} = P_i + Q_j\); and \(d_{ij}^2(b) = (x_i^A - x_j^B - b)^T S_{ij}^{-1} (x_i^A - x_j^B - b)\) is the squared Mahalanobis distance between tracks \(i\) and \(j\), parameterized by a bias estimate \(b\). It is also possible for track \(i \in N_A\) to be unassigned, in which case the incremental likelihood is the null assignment likelihood \(\beta_{NTA}\beta_{NTB}\), where \(\beta_{NTA} = \beta_T P_{AB}^k \beta_{FA}\) represents a target density of no target existing for sensor \(A\), and \(P_{AB}\) is the probability of tracking an object with sensor \(B\) but not with sensor \(A\): \(\beta_{NTB} = \beta_T P_{AB}^k + \beta_{FB}\) represents a target density of no target existing for sensor \(B\), and \(P_{AB}\) is the probability of tracking an object with sensor \(A\) but not with sensor \(B\); the densities \(\beta_{FA}\) and \(\beta_{FB}\) represent the false track densities for sensor \(A\) and \(B\), respectively. False tracks are not uncommon when tracking extended objects, i.e., objects for which a sensor may receive multiple detections on a given data frame.

Multiplying these likelihoods together, we arrive at the GNPM likelihood function:

\[
 L(b, j) = \frac{e^{-b^TR^{-1}b/2}}{(2\pi)^{D/2}/\sqrt{|R|}} \prod_{i=1}^{n_A} \left\{ \beta_T P_{AB}^k \frac{e^{-d_{ij}^2(b)/2}}{(2\pi)^{D/2}/\sqrt{|S_{ij}|}} \right\} \text{ if } j_i > 0 \right\} \text{ if } j_i = 0 } \right).
\]

From a computational perspective, it is more convenient to work with the negative log likelihood. After some algebra and the removal of unnecessary constants, we obtain a modified version of the negative log likelihood function \(NLL(b, j) = \) \[ b^T R^{-1} b + \sum_{i=1}^{n_A} \left\{ d_{ij}^2(b) + \log(|S_{ij}|) \right\} \text{ if } j_i > 0 \right\} \text{ if } j_i = 0 } \right)
\]

where

\[
 g = -2 \log \left( \frac{\beta_{NTA}\beta_{NTB}(2\pi)^{D/2}}{\beta_T P_{AB}} \right)
\]

is a so-called (log likelihood) gate value, which can be interpreted as a cost incurred for assigning a track \(i \in N_A\) to the dummy track \(j = 0\). Extensions for feature-aided association have been made (see, e.g., \([1]\)), but this topic lies beyond the scope of this work.

### 3 Derivation of Track-to-Track Association Likelihoods

In certain circumstances, we may wish to work with a “pure” track-to-track association likelihood, which we refer to simply as an association likelihood. In particular, we continue to assume that the relative bias is a Gaussian random vector, but rather than optimize the joint bias-association likelihood, we optimize only the marginal likelihood of track-to-track association. The purpose of isolating the likelihood solely in terms of a
track-to-track association is to make it possible to compute pairwise association likelihoods for track ambiguity management, as will be explained in the subsequent section. To this end, we remove the likelihood term for the relative bias in Equation (1) by integrating over all possible bias estimates. This yields

\[
L(j) = \int_{b \in \mathbb{R}^D} \prod_{i=1}^{n_A} \left\{ \begin{array}{ll}
\beta_T P_{AB} e^{-\frac{x_i^T(b-x_j)^2}{2(2\pi)^D/2 |S_i|}} & \text{if } j_i > 0 \\
\beta_{NTA} \beta_{NTB} & \text{if } j_i = 0
\end{array} \right\} e^{-b^T R^{-1} b/2} (2\pi)^{D/2/|R|} db.
\] (4)

We would like to show that for a given complete assignment \( j \), Equation (4) has a convenient closed-form solution. For simplicity, assume that \( \beta_T P_{AB} = 1 \), that \( j_i > 0 \) for all \( i \), and let \( x_i = x_i^A - x_i^B \) and \( S_i = P_i + Q_{ji} \), for \( i \in N_A \). Then, Equation (4) becomes

\[
L(j) = \int_{b \in \mathbb{R}^D} \prod_{i=1}^{n_A} e^{-\frac{1}{2}(x_i-b)^T S_i^{-1}(x_i-b)} e^{-b^T R^{-1} b/2} (2\pi)^{D/2/|S_i|} db.
\] (5)

Defining \( x_0 = 0 \) and \( S_0 = \mathbb{R} \), Equation (5) becomes

\[
L(j) = \int_{b \in \mathbb{R}^D} \prod_{i=0}^{n_A} f_{X_i}(b) db.
\] (6)

Notice that if we let \( X_i \) denote a multivariate Gaussian random variable with dimension \( D \), mean \( x_i \), and covariance \( S_i \), for \( i = 0, \ldots, n_A \), then Equation (6) is nothing more than the integral of the product of \( n_A + 1 \) Gaussian random variables

\[
L(j) = \int_{b \in \mathbb{R}^D} \prod_{i=0}^{n_A} f_{X_i}(b) db.
\]

It can be shown that

\[
L(j) = \frac{\sqrt{(2\pi)^D |V|}}{\prod_{i=0}^{n_A} (2\pi)^{D/2} |S_i|} e^{-\frac{1}{2} \zeta},
\]

where \( V = (\sum_{i=0}^{n_A} S_i^{-1})^{-1} \), \( \zeta = (\sum_{i=0}^{n_A} x_i^T S_i^{-1} x_i)^{-1} u^T V u \), and \( u = (\sum_{i=0}^{n_A} S_i^{-1} x_i) \).

Returning to the more general case in Equation (4), let \( I \) denote the set of sensor \( A \) tracks assigned to a sensor \( B \) track, i.e., \( I = \{ i \in N_A : j_i > 0 \} \) and let \( I^0 \) denote the set of unassigned sensor \( A \) tracks, i.e., \( I^0 = \{ i \in N_A : j_i = 0 \} \). Then, \( I^* = I \cup \{ 0 \} \). Then, \( L(j) = (\beta_{NTA} \beta_{NTB})^{I^0} (\beta_T P_{AB})^I \times \frac{\sqrt{(2\pi)^D |V|}}{\prod_{i \in I^*} (2\pi)^{D/2} |S_i|} e^{-\frac{1}{2} \zeta}, \) (7)

where \( V = (\sum_{i \in I^*} S_i^{-1})^{-1} \), \( \zeta = (\sum_{i \in I^*} x_i^T S_i^{-1} x_i)^{-1} u^T V u \), and \( u = (\sum_{i \in I^*} S_i^{-1} x_i) \).

**Implications for Track Ambiguity Management**

Having derived a closed-form expression for an association likelihood over all possible relative bias values, we are now in a position to describe how one can generate a confusion matrix of individual track pairing likelihoods. As argued in the introduction, possessing individual track pairing likelihoods can be beneficial at the system-level where inherent uncertainties make it difficult to rank one association of track sets over another. Fundamentally, individual track pairing likelihoods provide a system-level tracking and discrimination architecture a quantifiable level of confidence that certain objects of interest should be paired together.

Let \( j^1, \ldots, j^r \) denote all \( r \) possible track-to-track association vectors and \( L(j^1), \ldots, L(j^r) \) the corresponding likelihoods as computed in Equation (7). Let \( T_k \) denote the set of track pairings in the \( k \)th best association hypothesis, for \( k = 1, \ldots, r \). For all \((i, j)\) pairs of tracks with \( i = 0, \ldots, n_A \) and \( j = 0, 1, \ldots, n_B \), we can compute a pairwise likelihood

\[
L_{ij} = \frac{1}{L_N} \sum_{k=1}^{r} L(j^k),
\]

where \( L_N = \sum_{k=1}^{r} L(j^k) \) is a normalizing constant. Together these pairwise likelihoods form what is often called a confusion matrix.

An obvious intractability in the above calculation is that the correct pairwise likelihood \( L_{ij} \) and the correct normalizing likelihood require the explicit enumeration of all \( r \) possible track-to-track association hypotheses, which grows factorially in the number of tracks \( n_A \) and \( n_B \) as noted in [7] and [14]. Since exhaustive enumeration is all but impossible except when \( n_A \) and \( n_B \) are sufficiently small, a heuristic approach is to compute approximate pairwise likelihoods

\[
\hat{L}_{ij} = \frac{1}{L_N} \sum_{k=1}^{\hat{r}} L(j^k),
\]

where \( \hat{r} \) is the number of association hypotheses that are returned by an algorithm for solving the GNPM problem in a limited amount of time and \( \hat{L}_N = \sum_{k=1}^{\hat{r}} L(j^k) \) is an approximate normalizing constant.

Two remarks are in order. First, those familiar with multiple hypothesis tracking (MHT) might recognize that a similar heuristic approach for computing approximate track hypothesis likelihoods is commonly used in MHT since one cannot enumerate all possible detection-to-track associations (see, e.g., Chp. 16 of [2]). Second, note that it would be incorrect to use the bias-association likelihoods \( L(b, j) \) in computing the pairwise likelihoods \( L_{ij} \) since each \( L(b, j) \) is weighted by a unique relative bias probability. Since the relative
bias estimate differs from association to association, it does not make sense to compute pairwise likelihoods in this manner. Moreover, empirically we have found that even approximating pairwise likelihoods in this manner can lead to unsatisfactory results.

4 Optimizing with respect to Association Likelihood

Equipped with an association likelihood function that accounts for the presence of a random relative bias, it is natural to ask whether we can and should optimize track-to-track associations with respect to this likelihood function. In the last section, we attempted to provide a modestly compelling answer to the latter question regarding why one should at least consider using the association likelihood function for track-to-track association. In this section and the next, we answer the former question by describing how one can perform track-to-track association by optimizing with respect to the association likelihood function. In order to formalize this optimization problem, our objective in this section is to cast the track-to-track association problem, using the association likelihood function \( f \), as a mathematical program, specifically, as a 0-1 nonlinear optimization problem.

Starting from Equation (7), we collect the \((2\pi)^{D/2}\) terms and use the fact that \(|I| + |I'| = n_A\) to write

\[
L(j) = (\beta_{NTA} \beta_{NTB})^{n_A} \left( e^{g} \right)^{|I|} \frac{\sqrt{|V|}}{\sqrt{\prod_{i \in I^+} |S_i|}} e^{-\frac{1}{2} \kappa},
\]

where \( g \) is the gate value defined in Equation (3). Taking the logarithm and multiplying through by \(-2\) yields

\[
-2 \log L(j) = -2 \log(\beta_{NTA} \beta_{NTB})^{n_A} - |I|g - \log(|V|) + \sum_{i \in I^+} \log(|S_i|) + \zeta.
\]

Replacing \( S_0 \) with \( R \) and \( \zeta \) with \((\sum_{i \in I} x_i^T S_i^{-1} x_i) - u^T V u\), where \( V = (R^{-1} + \sum_{i \in I} S_i^{-1})^{-1} \), we obtain

\[
-2 \log L(j) = \kappa + |I|g - \log(|V|) + \sum_{i \in I} (x_i^T S_i^{-1} x_i + \log(|S_i|)) - u^T V u,
\]

where \( \kappa = -2 \log(\beta_{NTA} \beta_{NTB})^{n_A} - n_A g + \log(|R|) \).

We now introduce binary decision variables \( y_{ij} \) such that \( y_{ij} = 1 \) if sensor \( A \) track \( i \) is assigned to sensor \( B \) track \( j \) (or possibly the dummy track \( j = 0 \)), and \( y_{ij} = 0 \) otherwise. For compactness, it will be convenient to define the set \( \mathcal{Y} = \{ y \in \{0,1\}^{n_A \times (n_B+1)} : \sum_{j=0}^{n_B} y_{ij} = 1, \text{ for } i \in N_A; \sum_{i=1}^{n_A} y_{ij} \leq 1, \text{ for } j \in N_B \} \) as the set of all feasible associations. Finally, since \( \kappa \) is independent of the track-to-track association made, we treat it as a constant and remove it from the likelihood function when we optimize.

The problem of finding an optimal track-to-track association over all possible relative bias values can now be formulated as the following constrained optimization problem, which we will henceforth refer to simply as the TTA problem:

\[
\min_y \sum_{i=1}^{n_A} \sum_{j=0}^{n_B} c_{ij} y_{ij} - \log(|V|) - (Ay)^T V (Ay)
\]

\[
\text{s.t. } V = \left( R^{-1} + \sum_{i=1}^{n_A} \sum_{j=1}^{n_B} S_{ij}^{-1} y_{ij} \right)^{-1}
\]

\[
y \in \mathcal{Y}
\]

where \( y \in \{0,1\}^{n_A(n_B+1)} \) is a vectorized form of the \( y_{ij} \) variables, \( A \) is a \( D \times n_A(n_B+1) \) matrix of reals whose columns are such that

\[
Ay = \sum_{i=1}^{n_A} \sum_{j=0}^{n_B} \left\{ S_{ij}^{-1}(x_i^A - x_j^B) \right\} y_{ij},
\]

the coefficients

\[
c_{ij} = \left\{ \begin{array}{ll} d_{ij}^2(0) + \log |S_{ij}| & \text{if } j \in N_B \\ g & \text{if } j = 0 \end{array} \right.,
\]

for \( i \in N_A \), are given, and \( g \) is the gate value defined in Equation (3). The mathematical program (10) is known as a 0-1 nonlinear optimization problem, which is a class of optimization problems that is generally difficult to solve to optimality.

5 Solution Methods

In this section, we discuss solution techniques for solving the TTA problem. In general, the class of 0-1 nonlinear optimization problems is challenging computationally and there are very few commercially-available optimization software packages that can solve them. Instead, highly-tailored algorithms that exploit idiosyncrasies of the particular problem are required. The TTA problem is no exception. The effort needed to identify a provably optimal solution (i.e., a track-to-track association hypothesis) grows exponentially in the number of tracks to be associated.

On the other hand, finding near optimal solutions (and, perhaps, an optimal solution) without any guarantee on the solution quality, is frequently possible. It can be shown that under certain conditions [15], which often arise given realistic data, the GNPM problem discussed in Section 2 provides a close approximation to the TTA problem. As a consequence, one approach to generating near optimal solutions as quickly as possible is first to solve the GNPM problem by applying the multitstart local search heuristics described in [3] and [14]. Because these are heuristics, no optimality guarantee is provided. Nevertheless, these methods are extremely fast and have proven to be important given real-time
processing constraints. Essentially, these heuristics apply a fast iterative local search method from various starting points (hence the term “multistart”) and find good bias-association hypotheses, \((b,j)\) pairs, to the GNPM problem. To recover the “pure” association likelihood of association hypothesis \(j\), one can solve Equation (7).

Given that heuristics are already available for finding good association hypotheses, we turn to the question of solving the TTA problem exactly, i.e., identifying a provably optimal solution. Typical approaches for solving 0-1 nonlinear optimization problems exactly include branch-and-bound methods, cutting-plane algorithms, and hybrids of these two known as branch-and-cut methods. Another interesting approach discussed in [8, 16], which is worthy of further investigation, is based on a technique known as “lifting” for 0-1 optimization problems and has gained increasing attention in the optimization community.

Branch-and-bound methods have been developed for the GNPM problem [3, 14] and closely resemble the one described below. The key difference between these methods is the construction of a lower bound at each node. This is one of our main contributions and is discussed below. Readers interested in implementation details may wish to consult [3] and [12].

## A Branch-and-Bound Framework

A systematic way of finding an optimal solution or the \(K\) best solutions to the TTA problem is via a divide-and-conquer procedure known as branch-and-bound. In this approach, one creates a search tree consisting of nodes at varying depths, where each node represents a partial or complete association. Branch-and-bound methods pervade the field of discrete optimization and are discussed in virtually every introductory textbook on the subject (see, e.g., [12]). They are founded upon the idea of exploring nodes (or partial associations) in the search tree in an intelligent manner so that not all associations need be examined. The term branch refers to the manner in which partial associations are constructed, i.e., the partitioning of the solution space into smaller and smaller subproblems. The term bound refers to deterministic bounds that are computed during the search process, which can be used to prune partial associations that cannot possibly lead to (i.e., be a parent of) an optimal solution. All children of a pruned node are said to be implicitly enumerated.

Without loss of generality, we assume that sensor \(A\) tracks are to be associated in increasing order with sensor \(B\) tracks, i.e., first track \(1 \in N_A\) must be assigned, then track \(2 \in N_A\), and so on. The root node has no associations and is said to be at depth 0. From the root node, \(n_B + 1\) branches are created giving rise to \(n_B + 1\) nodes at depth 1, which represent the association of sensor \(A\) track 1 with all \(n_B\) sensor \(B\) tracks plus the null partial association \(\{(1,0)\}\). From each node at depth 1 with an association of the form \(\{(1,j_1)\}\) with \(j_1 \in N_B\), \(n_B\) branches are created. Each of these nodes has a partial association of the form \(\{(1,j_1),(2,j_2)\}\), where \(j_1 = 1,\ldots,n_B\) and \(j_2 \in \{0,1,\ldots,n_B\}\setminus\{j_1\}\). From the node with partial association \(\{(1,0)\}\), \(n_B + 1\) branches are created since sensor \(A\) track 2 can be associated with any track in \(N_B\) plus the dummy track. This branching continues until a depth \(n_A\) is reached or until all nodes have been implicitly enumerated.

Of course, a primary goal in branch-and-bound is to avoid having to expand a node by determining in advance if it has the potential of leading to an optimal solution or one of the \(K\) best solutions. If it can be deduced that no child node of a node that is being considered for expansion can be better than the best complete association(s) found thus far in the search process, known as the incumbent solution(s), then the node can be pruned. Pruning is essential when solving large problem instances because it reduces the time and memory needed to explore the search tree. In our solution approach, we advocate finding \(K\) good incumbent solutions quickly by first solving the GNPM problem and then launching branch-and-bound to prove the optimality of these solutions or to find better solutions.

### Semidefinite Programming to Compute Valid Lower Bounds at a Node

Since tight bounds on the potential of a node are crucial in facilitating this pruning process, we now describe how one can determine a lower bound on the objective function value in (10) of a node at depth \(d = 0,1,\ldots,n_A-1\) in the search tree. Recall that, for a node at depth \(d\), all \(y_{ij}\) decision variables have been fixed to 0 or 1, for \(i = 1,\ldots,d\) and \(j = 0,1,\ldots,n_B\), meaning that the first \(d\) sensor \(A\) tracks have been associated with a sensor \(B\) track or a dummy track. Assume that after determining the optimal values of the remaining \(y_{ij}\) variables, the objective function value in (10) is \(z\) at this node. We would like a way of quickly computing a lower bound \(z_{LB}\) for this node, such that \(z_{LB} \leq z\), to facilitate pruning.

As a first pass, it is tempting to relax the binary constraints \(y \in \{0,1\}\) to continuous variable bound constraints \(y \in [0,1]\) as is typically done for mixed-integer linear programming problems. However, this relaxation is not particularly helpful by itself since the resulting optimization problem is a continuous nonconvex optimization problem whose problem structure is no better than before. Consequently, we seek an alternative relaxation. The vexing term in the objective function (10) of the TTA problem is \((Ay)^T V (Ay)\), which at first glance appears to be quadratic in the decision vector \(y\), but is actually not since the matrix \(V\) is itself a function of \(y\). If we could replace \(V\) with some other symmetric positive definite matrix \(W\) such that

\[
(Ay)^T W (Ay) \geq (Ay)^T V (Ay) \tag{11}
\]
and
\[ \log(|W|) \geq \log(|V|), \] (12)
then we could solve the following relaxation of the TTA problem:
\[ \min_{y \in Y} \sum_{i=1}^{n_A} \sum_{j=0}^{n_B} c_{ij} y_{ij} - \log(|W|) - (Ay)^T W (Ay), \] (13)
which is known as a 0-1 quadratic programming problem. Also, note that the term \( \log(|W|) \) is independent of the decision vector \( y \) and can be removed from the objective function in (13). Although 0-1 quadratic programs are generally difficult to solve exactly, it turns out that they can be approximated well using a semidefinite programming (SDP) relaxation [5]. Of equal importance, semidefinite programs are a special class of convex optimization problems, which can be solved efficiently, i.e., in polynomial-time, using interior-point methods [13]. Below we construct an SDP relaxation of (13).

Following the derivation in Helmberg [5], consider the following 0-1 quadratic programming problem:
\[ \max_{x \in \{0,1\}^n} x^T B x = \max_{x \in \{0,1\}^n} B \cdot x x^T, \] (14)
where \( x \in \{0,1\}^n \) is a binary decision vector, \( B \in \mathbb{R}^{n \times n} \) is a symmetric matrix, and \( A \cdot B = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{ij} \) is the inner product of two square matrices \( A \) and \( B \). Note that \( x_i^2 = x_i \) since \( x_i \) is binary, \( \forall i = 1, \ldots, n \), so \( \text{diag}(x x^T) \in \{0,1\}^n \), where \( \text{diag}(M) \) denotes a vector of the diagonal entries of a square matrix \( M \). Also note that \( x x^T \) is a rank one matrix. We can replace/relax \( x x^T \) with a positive definite matrix \( X \), which gives rise to the following SDP relaxation of (14):
\[ \max_{X \succeq 0} B \cdot X \]
\[ \text{s.t.} \quad \left( \begin{array}{c} 1 \\ \text{diag}(X)^T \\ X \end{array} \right) \succeq 0 \]
where \( M \succeq 0 \) means that a square matrix \( M \) is positive semidefinite. Applying this result to (13) with the constant term \( \log(|W|) \) removed, we obtain the following SDP relaxation:
\[ z_{SDP} = \min_{Y,Y} \sum_{i=1}^{n_A} \sum_{j=0}^{n_B} c_{ij} y_{ij} - A^T W A \cdot Y \]
\[ \text{s.t.} \quad \left( \begin{array}{c} 1 \\ \text{diag}(Y)^T \\ Y \end{array} \right) \succeq 0 \]
\[ \text{diag}(Y) = y, \quad Y \succeq 0, \quad y \in \text{conv}(Y) \] (15)
where \( \text{conv}(Y) \) denotes the convex hull of \( Y \). The constraint \( \text{diag}(Y) = y \) links the decision variables in the positive semidefinite matrix \( Y \) with those in the association vector \( y \). Note that this SDP formulation allows the decision variables \( y_{ij} \) to take fractional values. Although associating a fraction of a sensor \( A \) track to a sensor \( B \) track is meaningless, the solution furnished by the SDP relaxation is of little interest to us, unless \( y \) is binary, since our sole objective is to obtain a valid lower bound on the objective function in (10).

We now show how to appropriately choose \( W \) for each node in the search tree. Suppose we wish to examine a node at depth \( d \in \{0,1,\ldots,n_A-1\} \) in the search tree. We refer to this node as a parent node because, if expanded, it will give rise to child nodes. Recall that at this particular parent, the first \( d \) sensor \( A \) tracks have been associated to a sensor \( B \) track or the dummy track. Equivalently, this means that the corresponding \( y_{ij} \) decision variables are fixed at 0 or 1, for \( i = 1,\ldots,d \) and \( j = 0,1,\ldots,n_B \). We associate with this parent node, the matrix
\[ V_d^{-1} = R^{-1} + \sum_{i=1}^{n_B} \sum_{j=1}^{n_B} S_{ij}^{-1} y_{ij}. \]
(Hence, \( V_0 = R \).) Similarly, associated with a child of this parent is the matrix
\[ V_{d+1}^{-1} = V_d^{-1} + S_{ij}^{-1} y_{ij}, \]
where \( i = d+1 \) and \( j \in \{0,1,\ldots,n_B\} \). The following proposition reveals a nice relationship between the matrices \( V_d \) and \( V_{d+1} \).

**Proposition:** For all \( d \in \{0,1,\ldots,n_A-1\} \), we have
\[ V_{d+1}^{-1} \succeq V_d^{-1} \]
and
\[ -\log |V_{d+1}| \geq -\log |V_d| \]

**Proof** Since the matrices \( S_{ij} \) and \( S_{ij}^{-1} \) are positive definite for all \( i \) and \( j \), it follows that \( V_{d+1}^{-1} = V_d^{-1} + S_{ij}^{-1} y_{ij} \succeq V_d^{-1} \) for \( d = 0,\ldots,n_A-1 \). Moreover,
\[ V_{d+1}^{-1} \succeq \]
\[ \iff \]
\[ |V_{d+1}| \leq |V_d| \]
\[ \iff \]
\[ \log |V_{d+1}| \leq \log |V_d| \]
\[ \iff \]
\[ -\log |V_{d+1}| \geq -\log |V_d| \]

where the second equivalence holds because the determinant of the sum of two positive definite matrices is always greater than the determinants of either of those matrices.

QED

Informally, the proposition tells us that the matrix \( V_{d+1}^{-1} \) corresponding to a child node can only become more positive definite than its parent if sensor \( A \) track \( d+1 \) is associated with a sensor \( B \) track \( j \in N_B \). The implication of the proposition is: By setting \( W = V_d \) in (15), properties (11) and (12) are satisfied for all \( y_{ij} \) that are not fixed. Therefore, \( z_{LB} = z_{SDP} - \log(|V_d|) \) is a valid lower bound for the optimal objective function value of a node at depth \( d \) in the search tree.
In summary, our branch-and-bound procedure proceeds according to the description given in the introduction of this section, where a parent node is expanded to produce child nodes only if the lower bound \( z_{LB} \) computed for the parent via the SDP relaxation just outlined is below the objective function value (the likelihood) of the \( K \)th best solution found thus far in the search. Empirically, we have found that the SDP bounds are quite loose at depths \( d < n_A / 5 \), but become increasingly tight as more tracks are associated.

6 Conclusions

The primary goal of this paper was to introduce a “pure” track-to-track association likelihood function for track ambiguity management, which takes into account all of the major issues as other popular association likelihood functions, but is more suitable for system-level track ambiguity management. After making a case regarding why our association likelihood is useful in the missile defense domain, we described how pairwise track-to-track likelihoods could be constructed to quantify the confidence in pairing two tracks together. Our final contribution was an algorithm that could solve a track-to-track association problem using the likelihood function we introduced. Specifically, we proposed a branch-and-bound approach for solving our TTA problem exactly using semidefinite programming relaxations at each node to provide increasingly better lower bounds.

Acknowledgements

We wish to thank AJ Noushin, JD Sergi, and Andy Stachyra of Raytheon for valuable feedback. The first author would like to thank Maxim Raykin of Raytheon for helping recognize the closed-form solution in Equation (7) as well as Guanghui Lan and Renato Monteiro of Georgia Tech for instructive comments and references concerning convex and semidefinite programming.

References


