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

DIMITRI J. PAPAGEORGIOU
MICHAEL HOLENDER

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 marginal association likelihoods. Finally, we provide two algorithms that find the K provably best track-to-track associations with respect to our new likelihood function. These results facilitate what is commonly known as system-level track ambiguity management.

Manuscript received June 25, 2010; revised January 25, 2011; released for publication April 26, 2011.

Refereeing of this contribution was handled by Dr. Stefano Coraluppi.

The first author was partially supported by a National Science Foundation Graduate Research Fellowship.

Authors addresses: D. J. Papageorgiou, H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, 765 Ferst Drive NW, Atlanta, GA 30332, E-mail: (djpapag@gatech.edu); M. Holender, The Raytheon Company, Integrated Defense Systems, 225 Presidential Way, Woburn, MA 01801, E-mail: (Michael_N_Holender@raytheon.com).

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 [10] model encompasses many different processes while attempting the overall goal of Object Refinement. Raw data are input into the system and the first problem that is considered is determining what that data refer 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.

A typical multiple target tracking (MTT) system is composed 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 [21].

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 [3, 19, 20].

As discussed in [3], [11], and [19], there are several potential sources of error that bias removal algorithms attempt to rectify. Moore and Blair [19] group these sources into three broad categories: sensor errors, sensor/platform position and heading errors, and transformation errors from one sensor to another. Sensor registration attempts to estimate errors associated with biases that are constant or changing very slowly with time so that they can be removed before filtering takes place. Sensor biases, which may arise in both range and angle dimensions, often account for the majority of the total error. Moreover, they are typically the most difficult to estimate and eliminate. While range error can consist of offset and scaling errors, the offset error is typically the culprit of the error that sensor registration seeks to remove. In this paper, we only consider the presence of

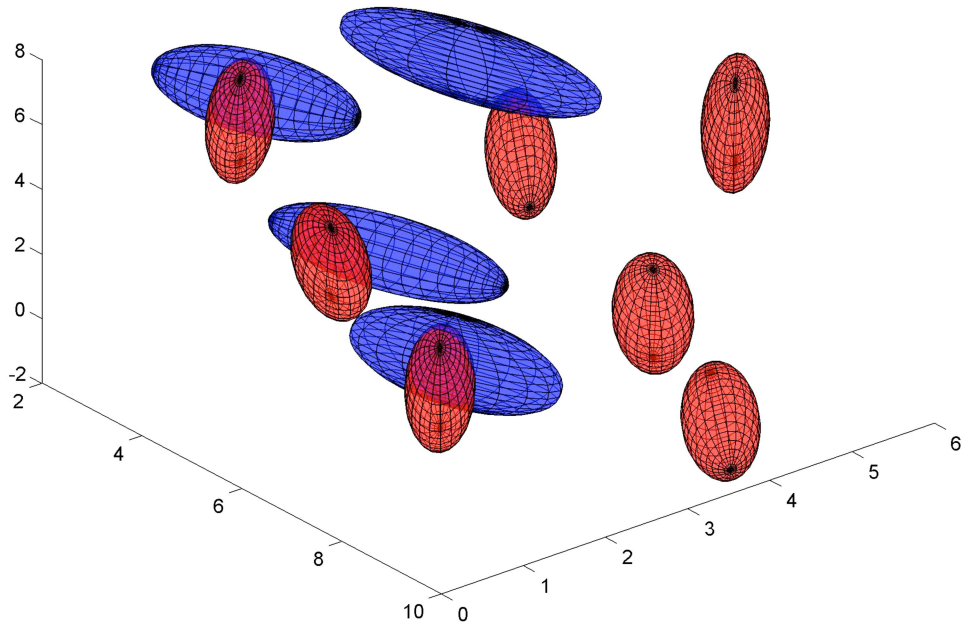


Fig. 1. The track-to-track association problem using positional densities only.

a random relative bias between sensors, without the aid of so-called *targets of opportunity* [15] or absolute bias estimates [12].

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. Fig. 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.

Our point of departure is a track-to-track association problem posed by Levedahl [17], 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 [14] 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., [27]). 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. A precise description of the GNPM problem is given in Section 2.

Algorithms that solve the GNPM problem typically generate the K best bias-association hypotheses and rank them according to their corresponding bias-association likelihood. One of the questions that we hope to address is: Is this information adequate for helping a system operator overseeing not just a suite of sensors, but weapon systems and other technology as well, make a more informed decision concerning whether or not to fuse the track data? In the air and missile defense domain, system-level tracking carries inherent uncertainties as heterogeneous sensors with different viewing geometries that 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 per-

spective 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 paper, we argue that the information returned after solving the GNPM problem for track-to-track association may be insufficient in assisting a system operator make an informed decision concerning whether or not two tracks should be fused, especially in the case when a one-time sensor-to-sensor handover is necessary. Instead, we suggest that a closely-related, but different, likelihood function, which we call a *marginal track-to-track association likelihood function*, can be used either in lieu of the GNPM likelihood function or for the computation of individual track pairing likelihoods to improve system-level ambiguity management.

Our main contributions to the fusion community are the following: (1) We derive a closed-form solution for our proposed likelihood function and provide a procedure for computing likelihoods for individual track pairs. (2) We illustrate the usefulness of these track pair likelihoods with a detailed example based on four track scenes of increasing ambiguity. (3) We propose two exact algorithms for identifying the K provably best solutions to our new association problem, one of which may be suitable within an operational system.

The outline of this paper is as follows: In Section 2, a common set of assumptions is given and the GNPM problem is presented. In Section 3, we formally derive a closed-form expression for our marginal track-to-track association likelihood and contrast it with the GNPM likelihood function. A detailed example then follows of the practicality of the marginal track-to-track association likelihood function for system-level track ambiguity management. In Section 5, motivated by the findings presented in the example of Section 4, we propose an alternative formulation for performing track-to-track association using our marginal track-to-track association likelihood function in lieu of the GNPM likelihood function and two exact algorithms for solving this problem in Section 6. Computational results are presented in Section 7 followed by conclusions in Section 8.

2. PROBLEM DESCRIPTION

Consider two independent sensor systems—sensor A and sensor B—tracking an unknown number of targets in space. Let $N_A = \{1,\ldots,n_A\}$ and $N_B = \{1,\ldots,n_B\}$ denote the set of tracks formed by sensors A and B, respectively. Without loss of generality, we assume throughout that $n_A \leq n_B$. Due to a host of factors, including geometry and sensor resolution, the number of tracks formed by sensors 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 ith 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 jth sensor B track. We assume that estimation errors for each sensor reporting on a common target are uncorrelated. This is only an approximation since, in general, track errors from different sensors are correlated. We assume all state estimates and covariance matrices have been extrapolated to a common time point and have been converted to a common D-dimensional reference frame.[1]

A key assumption in this track-to-track association framework is that each track state is corrupted by a constant, but unknown, sensor bias[2] [5, 17, 21, 25]. Ideally, these individual sensor biases would be estimated and removed prior to performing track-to-track association, but this is not always possible [3, 19]. 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. That is, this is a Bayesian estimation framework. The relative bias $\mathbf{b}$ is modeled as a Gaussian random vector having mean $\mathbf{0}$ and covariance $\mathbf{R}$ in a Cartesian coordinate frame. It is assumed that sensor bias only degrades a sensor's capability of measuring target state, and not its ability to detect a target.

We denote a track-to-track association by the vector $\mathbf{j}$. Consequently, the association of the ith track in N_A with the jth track in N_B is denoted by (i, j_i). It is convenient to think of the pair (i, j) as an undirected arc in a bipartite graph and the vector $\mathbf{j}$ as a compact notation for writing $\{(1, j_1),(2, j_2),\ldots,(n_A, j_{n_A})\}$. It is possible that the ith 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. We sometimes refer to partial and complete assignments. A *partial* assignment is one in which a strict subset of the sensor A tracks are assigned, while in a *complete* assignment, all sensor A tracks are assigned. A partial assignment can be made complete by assigning the currently unassigned sensor A tracks to the dummy track. It is implicitly assumed that at

[1] All vectors are column vectors. All vectors and matrices are written in bold font. All covariance matrices are assumed positive definite.

[2] It is worth noting that the assumption that average biases in synchronized data, transformed to a common reference frame, are equal is seldom true, even for identical sensors, unless they are collocated. This is the case for radars because during the process of spatial transformation to a common coordinate frame, the two biases get magnified nonlinearly and differently, as the latitudes, longitudes, and the ECR coordinates of the two sensors are different. Hence, this assumption is an approximation.

most one sensor A track can be assigned to a sensor B track and vice versa. We refer to the pair $(\mathbf{b},\mathbf{j})$ 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. Here, we follow the derivation of the likelihood function given in [17]. A general derivation of the *a posteriori* joint-probability-mass-density mixture function for more than two sensors is given in [16] before being specialized to the case of two sensors. 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

$$\frac{e^{-\mathbf{b}^T\mathbf{R}^{-1}\mathbf{b}/2}}{(2\pi)^{D/2}\sqrt{|\mathbf{R}|}}$$

(where $|\mathbf{R}|$ denotes the determinant of $\mathbf{R}$) is nothing more than a prior probability density on the relative bias, which we assume is available. We refer the reader to [5] for further discussion on the construction of priors in this framework. The second term consists of the product of the incremental likelihoods of track assignment. Specifically, given a bias estimate $\mathbf{b}$, the likelihood of assigning track $i \in N_A$ and track $j \in N_B$ is

$$\beta_T P_{AB} \frac{e^{-d_{ij}^2(\mathbf{b})/2}}{(2\pi)^{D/2}\sqrt{|\mathbf{S}_{ij}|}}$$

where

- β_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;
- $\mathbf{S}_{ij} = \mathbf{P}_i + \mathbf{Q}_j$;
- $d_{ij}^2(\mathbf{b}) = (\mathbf{x}_i^A - \mathbf{x}_j^B - \mathbf{b})^T\mathbf{S}_{ij}^{-1}(\mathbf{x}_i^A - \mathbf{x}_j^B - \mathbf{b})$ is the squared Mahalanobis distance between tracks i and j, parameterized by a bias estimate $\mathbf{b}$.

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

- $\beta_{NTA} = \beta_T P_{A\bar{B}} + \beta_{FA}$ represents a target density of no target existing for sensor A, and $P_{A\bar{B}}$ is the probability of tracking an object with sensor B but not with sensor A;
- $\beta_{NTB} = \beta_T P_{\bar{A}B} + \beta_{FB}$ represents a target density of no target existing for sensor B, and $P_{\bar{A}B}$ is the probability of tracking an object with sensor A but not with sensor B;
- the densities β_{FA} and β_{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*:

$$L(\mathbf{b},\mathbf{j}) = \frac{e^{-\mathbf{b}^T\mathbf{R}^{-1}\mathbf{b}/2}}{(2\pi)^{D/2}\sqrt{|\mathbf{R}|}} \times \prod_{i=1}^{n_A} \left\{ \begin{array}{ll} \beta_T P_{AB} \dfrac{e^{-d_{ij_i}^2(\mathbf{b})/2}}{(2\pi)^{D/2}\sqrt{|\mathbf{S}_{ij_i}|}} & \text{if} \quad j_i > 0 \\ \beta_{NTA}\beta_{NTB} & \text{if} \quad j_i = 0 \end{array} \right\}. \tag{1}$$

It should be noted that we, like many others, abuse terminology by referring to $L(\mathbf{b},\mathbf{j})$ as a likelihood function, when, in fact, it is a posterior joint-probability-mass-density mixture function as in Corollary 1 in [16], although our $L(\mathbf{b},\mathbf{j})$ differs by a factor from that given in Corollary 1 in [16]. Note that we have assumed that assignment likelihoods for track pairs are independent. 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

$$-\log L(\mathbf{b},\mathbf{j}) = \mathbf{b}^T\mathbf{R}^{-1}\mathbf{b} + \sum_{i=1}^{n_A} c_{ij_i}(\mathbf{b}) \tag{2}$$

where

$$c_{ij}(\mathbf{b}) = \begin{cases} d_{ij}^2(\mathbf{b}) + \log|\mathbf{S}_{ij}| & \text{if} \quad j \in N_B \\ g & \text{if} \quad j = 0 \end{cases}$$

and

$$g = -2\log\left(\frac{\beta_{NTA}\beta_{NTB}(2\pi)^{D/2}}{\beta_T P_{AB}}\right) \tag{3}$$

is the so-called (log likelihood) *gate value*, which can be interpreted as a cost incurred for assigning the ith sensor A track to the dummy track $j = 0$. Extensions for feature-aided association have been made (see, e.g., [2, 7, 27]), but this topic lies beyond the scope of this work.

Having characterized the fundamentals of how a bias-association hypothesis is "scored" via the GNPM likelihood function, we now explicitly formulate the GNPM problem as a mathematical program, specifically as a mixed-integer nonlinear program (MINLP) [5, 25]. This formulation will be necessary in Section 5 as we contrast it with an alternative formulation. The standard assumption in track-to-track association, which we follow, is that each sensor A track can be assigned to at most one sensor B track and vice versa. After introducing binary decision variables y_{ij}, for $i = 1,\ldots,n_A$ and for $j = 0,\ldots,n_B$, such that y_{ij} takes value one if sensor A track i is assigned to sensor B track j (or possibly the dummy track $j = 0$), and is zero otherwise, we can now cast the *GNPM problem* as the following constrained

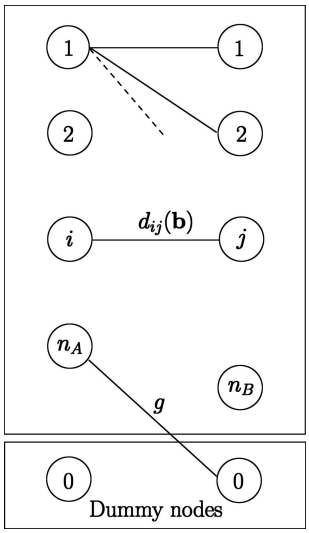


Fig. 2. Undirected bipartite graph.

minimization problem:

$$\begin{aligned} \min \quad & \mathbf{b}^T\mathbf{R}^{-1}\mathbf{b} + \sum_{i=1}^{n_A}\sum_{j=0}^{n_B} c_{ij}(\mathbf{b})y_{ij} \\ \text{s.t.} \quad & \mathbf{y} \in \mathcal{Y}, \quad \mathbf{b} \in \mathbb{R}^D \end{aligned} \tag{4}$$

where $\mathcal{Y} = \{\mathbf{y} \in \{0,1\}^{n_A \times (n_B+1)} : \sum_{j=0}^{n_B} y_{ij} = 1, \text{ for } i \in N_A, \sum_{i=1}^{n_A} y_{ij} \leq 1, \text{ for } j \in N_B\}$ is the set of all feasible track-to-track associations and $\mathbf{y}$ is a vectorized form of the y_{ij} variables. The above notation is standard in mathematical programming.

Basic characteristics and important observations of the GNPM problem are provided in [25]. Although Danford et al. [5] offer a similar formulation which they classify as a modified general network flow problem, we prefer to think of the GNPM problem as a two-dimensional nonlinear assignment problem, which is an extension of the traditional two-dimensional linear assignment problem used to perform track-to-track association. For those unfamiliar with mathematical programming, it may be convenient to interpret the GNPM problem (4) as a matching problem on a bipartite graph, depicted in Fig. 2, in which the objective is to minimize the sum of total arc costs subject to the constraint that each sensor A track can be assigned to at most one sensor B track and vice versa, where the arc costs are a function of the relative bias vector $\mathbf{b}$, and, hence, are nonlinear in the decision variables. Note that in Fig. 2 each non-dummy node on the left is connected to each non-dummy node on the right with a cost of $c_{ij}(\mathbf{b})$. Each dummy node is also connected to every node on the opposite side with a cost of g.

3. DERIVATION OF A MARGINAL TRACK-TO-TRACK ASSOCIATION LIKELIHOOD

In certain circumstances, we may wish to rank association hypotheses with respect to a "pure" track-to-track association likelihood, which we refer to as a *marginal track-to-track association likelihood* or MTTA likelihood for short. 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. In addition to reporting MTTA likelihoods to a system-user, another purpose of isolating the likelihood solely in terms of a track-to-track association is to facilitate the computation of pairwise association likelihoods for track ambiguity management, as will be explained below. After the initial release of this paper, we learned that Ferry [9] also advocates using MTTA likelihoods, which he calls *exact association probabilities*, and provides a comprehensive derivation of what follows for the more general setting involving more than two sensors and feature data.

To obtain our desired MTTA likelihood function, we remove the likelihood term for the relative bias in Equation (1) by integrating over all possible bias estimates. This yields $L(\mathbf{j}) =$

$$\int_{\mathbf{b}\in\mathbb{R}^D} \prod_{i=1}^{n_A} \left\{ \begin{array}{ll} \beta_T P_{AB} \dfrac{e^{-d_{ij_i}^2(\mathbf{b})/2}}{(2\pi)^{D/2}\sqrt{|\mathbf{S}_{ij_i}|}} & \text{if} \quad j_i > 0 \\ \beta_{NTA}\beta_{NTB} & \text{if} \quad j_i = 0 \end{array} \right\} \times \frac{e^{-\mathbf{b}^T\mathbf{R}^{-1}\mathbf{b}/2}}{(2\pi)^{D/2}\sqrt{|\mathbf{R}|}} d\mathbf{b}. \tag{5}$$

We would like to show that for a given complete assignment $\mathbf{j}$, Equation (5) has a convenient closed-form solution. For simplicity, assume that $\beta_T P_{AB} = 1$, that $j_i > 0$ for all i, and let $\mathbf{x}_{ij_i} = \mathbf{x}_i^A - \mathbf{x}_{j_i}^B$ and $\mathbf{S}_{ij_i} = \mathbf{P}_i + \mathbf{Q}_{j_i}$, for $i \in N_A$. Then, Equation (5) becomes $L(\mathbf{j}) =$

$$\int_{\mathbf{b}\in\mathbb{R}^D} \prod_{i=1}^{n_A} \frac{e^{-(1/2)(\mathbf{x}_{ij_i}-\mathbf{b})^T\mathbf{S}_{ij_i}^{-1}(\mathbf{x}_{ij_i}-\mathbf{b})}}{(2\pi)^{D/2}\sqrt{|\mathbf{S}_{ij_i}|}} \frac{e^{-\mathbf{b}^T\mathbf{R}^{-1}\mathbf{b}/2}}{(2\pi)^{D/2}\sqrt{|\mathbf{R}|}} d\mathbf{b}. \tag{6}$$

Defining $\mathbf{x}_{0j_0} = \mathbf{0}$ and $\mathbf{S}_{0j_0} = \mathbf{R}$, Equation (6) becomes

$$L(\mathbf{j}) = \int_{\mathbf{b}\in\mathbb{R}^D} \prod_{i=0}^{n_A} \frac{e^{-(1/2)(\mathbf{b}-\mathbf{x}_{ij_i})^T\mathbf{S}_{ij_i}^{-1}(\mathbf{b}-\mathbf{x}_{ij_i})}}{(2\pi)^{D/2}\sqrt{|\mathbf{S}_{ij_i}|}} d\mathbf{b}. \tag{7}$$

Notice that if we let X_i denote a multivariate Gaussian random variable with dimension D, mean $\mathbf{x}_{ij_i}$, and covariance $\mathbf{S}_{ij_i}$, for $i = 0, \ldots, n_A$, then Equation (7) is nothing more than the integral of the product of $n_A + 1$ independent Gaussian random variables

$$L(\mathbf{j}) = \int_{\mathbf{b}\in\mathbb{R}^D} \prod_{i=0}^{n_A} f_{X_i}(\mathbf{b}) d\mathbf{b}.$$

It can be shown that

$$L(\mathbf{j}) = \frac{\sqrt{(2\pi)^D|\mathbf{V}|}}{\sqrt{\prod_{i=0}^{n}(2\pi)^D|\mathbf{S}_{ij_i}|}} e^{-(1/2)\zeta}$$

where $\mathbf{V} = (\sum_{i=0}^{n_A}\mathbf{S}_{ij_i}^{-1})^{-1}$, $\zeta = (\sum_{i=0}^{n_A}\mathbf{x}_{ij_i}^T\mathbf{S}_{ij_i}^{-1}\mathbf{x}_{ij_i}) - \mathbf{u}^T\mathbf{V}\mathbf{u}$, and $\mathbf{u} = \sum_{i=0}^{n_A}\mathbf{S}_{ij_i}^{-1}\mathbf{x}_{ij_i}$ (see [23] or [9]).

Returning to the more general case in Equation (5), 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., $I^0 = \{i \in N_A : j_i = 0\}$. Let $I^+ = I \cup \{0\}$. Then,

$$L(\mathbf{j}) = (\beta_{NTA}\beta_{NTB})^{|I^0|}(\beta_T P_{AB})^{|I|} \times \frac{\sqrt{(2\pi)^D|\mathbf{V}|}}{\sqrt{\prod_{i\in I^+}(2\pi)^D|\mathbf{S}_{ij_i}|}} e^{-(1/2)\zeta} \quad (8)$$

where $\mathbf{V} = (\sum_{i\in I^+}\mathbf{S}_{ij_i}^{-1})^{-1}$, $\zeta = (\sum_{i\in I^+}\mathbf{x}_{ij_i}^T\mathbf{S}_{ij_i}^{-1}\mathbf{x}_{ij_i}) - \mathbf{u}^T\mathbf{V}\mathbf{u}$, and $\mathbf{u} = \sum_{i\in I^+}\mathbf{S}_{ij_i}^{-1}\mathbf{x}_{ij_i}$. Note that in the remainder of the paper, when we refer to the *MTTA likelihood function* we typically have Equation (8) in mind since it is in closed-form, but one can also think of Equation (5) since this is the typical form of a marginal density function.

3.1. Implications for Track Ambiguity Management

Track-to-track association is one of many components in a multi-sensor MTT system. The decision logic used to create a single integrated air picture from multi-sensor data can vary widely from system to system, battle manager to battle manager. For this reason, modern association algorithms are being asked to return quantities besides the single most likely association to facilitate track ambiguity management, i.e., to assist the highest-level decision maker in managing track ambiguities. In this paper, we contend that the following information should often be included in the output returned by such an algorithm: (1) The K best GNPM bias-association solutions, (2) the K best MTTA solutions, and (3) a table of individual track pairing likelihoods or probabilities.

Having derived a closed-form expression for a marginal 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 $\mathbf{j}^1,\ldots,\mathbf{j}^r$ denote all r possible track-to-track association vectors and $L(\mathbf{j}^1),\ldots,L(\mathbf{j}^r)$ the corresponding likelihoods as computed in Equation (8). Let T_k denote the set of track pairings in the kth best association hypothesis, for $k = 1,\ldots,r$. For all (i,j) pairs of tracks with $i = 0,1,\ldots,n_A$ and $j = 0,1,\ldots,n_B$, we can compute a pairwise likelihood

$$L_{ij} = \frac{1}{L_N}\sum_{\substack{k=1:\\(i,j)\in T_k}}^{r} L(\mathbf{j}^k)$$

where $L_N = \sum_{k=1}^{r} L(\mathbf{j}^k)$ 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 [17] and [25]. 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

$$\hat{L}_{ij} = \frac{1}{\hat{L}_N}\sum_{\substack{k=1:\\(i,j)\in T_k}}^{\hat{r}} L(\mathbf{j}^k) \quad (9)$$

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

Three remarks are in order. First, track-to-track likelihoods L_{ij} could also be computed using techniques employed in the joint probabilistic data association (JPDA) method for calculating track-to-measurement probabilities (see, e.g., [28]). The computational complexity of these methods is exponential rather than factorial. We have suggested the heuristic above since it does not require any additional computation once the K best MTTA solutions have been found, which are useful to the battle manager in their own right. Second, 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 [3]). Third, note that it would be incorrect to use the bias-association likelihoods $L(\mathbf{b},\mathbf{j})$ in computing the pairwise likelihoods L_{ij} since each $L(\mathbf{b},\mathbf{j})$ 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 and others (see, e.g., Ferry [8]) have found that even approximating pairwise likelihoods in this manner can lead to unsatisfactory results.

3.2. Comparison of the MTTA and GNPM Likelihood Functions

The purpose of this subsection is to answer the following question: When might one prefer the MTTA likelihood function over the GNPM likelihood function? Below we provide three possible answers.

The first reason why one might favor the MTTA likelihood function over the GNPM likelihood function is a matter of interpretation. Recall that the fundamental objective of track-to-track association is to deter-

mine which sensor-level tracks correspond to the same true target from which the sensor-level tracks originated. Those who interpret this question to mean "What track-to-track association has the highest likelihood when weighted by the best relative bias estimate for that association?" are implicitly favoring the GNPM likelihood function in which the contribution of the best relative bias estimate for each association is used to compute the likelihood. Equivalently, one is implicitly placing a nontrivial emphasis on the relative bias estimation problem as they are on the association problem. On the other hand, those who interpret the fundamental question as "What track-to-track association has the highest likelihood over all possible relative bias values?" are implicitly placing a primary focus on the association problem and a secondary (minimal) focus on the relative bias estimation problem.

The second reason, which is closely related to the first, is to eschew overconfidence in any one particular association hypothesis. In certain instances, the difference between the GNPM likelihood of the best and next best bias-association hypotheses is large, even though the relative bias estimate for the best hypothesis is extremely unlikely. In some cases, this large difference can be misleading when deciding which tracks to fuse. This phenomenon can occur when the track-to-track association likelihood component of the joint likelihood function is large enough to offset the small likelihood of the corresponding relative bias estimate, for example, when many tracks with small track errors align nearly perfectly after an extremely unlikely relative bias is chosen. On the other hand, when all relative bias estimates are considered, the difference between the best and next best hypothesis is often less pronounced, and the ordering of the best hypotheses is often different, as will be shown in Section 4. In this case, a decision logic which uses association probabilities will perceive the bias-association ordering as being overly confident in the best bias-association hypothesis. See Ferry [8, 9] for related concerns.

The third reason why the MTTA likelihood function may be preferable is to facilitate the computation of individual track pairing likelihoods. Specifically, after computing the K best MTTA hypotheses, the calculation of approximate track pairing likelihoods requires virtually no additional computational effort. Since the goal of computing pairwise track-to-track association likelihoods implicitly emphasizes the association aspect, it seems natural to use the MTTA likelihood function. At the same time, it does not make mathematical/probabilistic sense to use bias-association likelihoods since each likelihood is weighted by a unique bias estimate that is different from hypothesis to hypothesis.

In summary, we are not suggesting that the marginal track-to-track association likelihood function presented in this paper is universally "better" than the GNPM likelihood function. However, we do advocate that it should be used to compute pairwise track-to-track association likelihoods and that it is a viable option when performing track-to-track association. Indeed, it may be more appropriate in certain circumstances in which one is emphasizing the association aspect of the fusion problem.

4. AN ILLUSTRATIVE EXAMPLE

The goal of this section is to present four track scenes illustrating (1) the differences between the GNPM and the MTTA likelihood functions, and (2) the benefits of the MTTA likelihood function in generating pairwise track-to-track association likelihoods for aiding system-level track ambiguity management. Specifically, we draw attention to the way in which pairwise association likelihoods can assist a system-user make an informed decision concerning whether or not to fuse two tracks.

Fig. 3 depicts four track-to-track association problem instances (track scenes) of increasing difficulty. Only three-dimensional positions are estimated and positional error baskets are shown. Our goal is to associate the four sensor A tracks (in blue) with the sensor B tracks (in red). Assume that the cost of a null assignment is relatively large, but not astronomical, so as to encourage actual track-to-track assignments. Scene 1 (upper left) is very easy as there are four sensor A tracks and four sensor B tracks. The correct association is identifiable by the eye. Scene 2 (upper right) has a nontrivial relative bias present; however, it is still easy to identify which tracks should be paired together. Scene 3 (lower left) has three tracks which can easily be associated, but there are two sensor B tracks in the upper right corner of this scene that could be paired with the fourth sensor A track in the same region of the figure. It is quite realistic that such a track scene can arise within certain domains of interest. For example, in ballistic missile tracking, a split track could occur in which sensor A collects detections during a given phase of the trajectory of a missile and forms four tracks. At some later point in time, sensor B begins observing the missile and forms five tracks because one of the objects splits into two after leaving sensor A's field of view. Scene 4 (lower right) is ambiguous. In one case, the four blue tracks could be paired with their neighboring red tracks (the ones touching the outer edge of the ellipsoids). In another case, the blue tracks 2, 3, and 4 could be paired with the red tracks 5, 6, and 7, respectively, towards the back right of the cube. We will see later that the former association is preferred.

Associated with each track scene are two tables. The first table lists the hypothesis number, the association vector $\mathbf{j}$, the negative log GNPM likelihood $-\log L(\mathbf{b},\mathbf{j})$ as computed in Equation (2), and the negative log MTTA likelihood $-\log L(\mathbf{j})$, as would be computed by taking the negative log of Equation (5), corresponding to the top twenty hypotheses that would be returned by an algorithm for the GNPM problem. Note that these hypotheses are ordered in nondecreasing order accord-

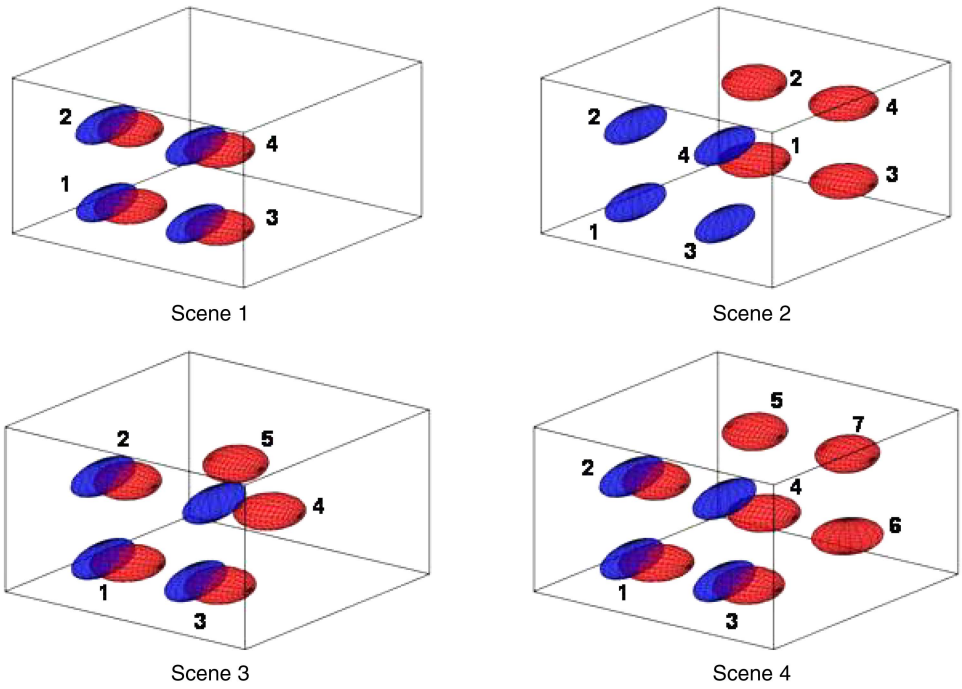


Fig. 3. Track scenes.

ing to $-\log L(\mathbf{b},\mathbf{j})$. As there are four track scenes, this information is presented in Tables I, III, V, and VII. The association vector is read as follows: Since there are four sensor A tracks in each scene, the assignment (j_1,j_2,j_3,j_4) means that sensor A track i is associated with sensor B track j_i, for $i = 1,\ldots,4$. If j_i is 0, then sensor A track i is unassigned (assigned to the dummy track). For example, the thirteenth best hypothesis (hypothesis 13) for track scene 1 has the assignment vector $(0,2,1,4)$, which means that sensor A track 1 is unassigned, while sensor A tracks 2, 3, and 4 are assigned to sensor B tracks 2, 1, and 4, respectively.

The second table associated with each track scene provides pairwise track-to-track association likelihoods as computed in Equation (9). Specifically, in Tables II, IV, VI, and VIII, one can look up in row i column j the likelihood that sensor A track i should be associated with sensor B track j. Similarly, the likelihood that sensor A track i should be unassociated (associated with a dummy track) is listed in row i and the column labeled 0. Note that in each row and column with a non-zero index the individual likelihoods sum to one. Although the pairwise likelihoods presented in these particular tables were computed using only the top twenty hypotheses, the pairwise likelihoods computed using all hypotheses gave the same result to six decimal places. As a matter of convenience, in the descriptions given for Tables I, III, V, and VII, the term "likelihood" should be interpreted to mean the negative log likelihood.

Track Scene 1: Very Easy

Table I provides the results for the K best ($K = 20$) hypotheses ordered with respect to the GNPM likelihood $-\log L(\mathbf{b},\mathbf{j})$. Notice that the first hypothesis has a significantly smaller negative log likelihood (hence, larger likelihood) than the next. This makes intuitive sense given that this particular scenario is very easy due to a small relative bias and a similar spatial configuration of the tracks from each sensor. Also, note that the ordering of the hypotheses would be slightly different if the MTTA likelihood $-\log L(\mathbf{j})$ were considered in lieu of the GNPM likelihood. This occurs throughout each of the four cases that we present and motivates the remaining sections of the paper.

Using the MTTA likelihoods $L(\mathbf{j})$, Table II was created to help the system-user understand the pairwise associations between tracks. As described in Section 3.1, one simply adds each $L(\mathbf{j})$ in the K-best hypotheses table for which the desired pairwise association exists and divide that value by the sum of all $L(\mathbf{j})$ values in the same table. In this example, the entries along the diagonal of Table II each achieve a high value of approximately 0.95 since the top hypothesis $\{(1,1),(2,2),(3,3),(4,4)\}$ is so much more likely than any other.

In considering a real-life fusion system, the end-user would find both of these tables useful in terms of assessing the current situation. The first table provides a list of complete track associations along with their relative likelihood values. This enables the user to consider all sensor tracks together. In addition, the user can utilize the second table to obtain information regarding individual track-to-track pairings. This can prove highly useful in a situation wherein the top two hypotheses are very close in terms of likelihood. In this case, it is certainly possible that the individual track-to-track associ-

TABLE I
Scene 1—Top 20 Bias-Association Hypotheses

Hyp	Assignment				$-\log L(\mathbf{b},\mathbf{j})$	$-\log L(\mathbf{j})$
1	1	2	3	4	15.6093	42.5388
2	1	0	3	4	25.0251	45.9716
3	1	2	3	0	25.0251	45.9717
4	0	2	3	4	25.0254	45.9717
5	1	2	0	4	25.0267	45.9722
6	2	1	3	4	28.1048	48.7875
7	1	2	4	3	28.1098	48.7890
8	1	4	3	2	28.1177	48.7936
9	3	2	1	4	28.1486	48.8081
10	1	0	3	2	29.3494	48.1344
11	2	0	3	4	29.3654	48.1421
12	1	2	4	0	29.3679	48.1430
13	0	2	1	4	29.3946	48.1560
14	0	1	3	4	29.8910	48.4052
15	3	2	0	4	29.8935	48.4055
16	1	2	0	3	29.8949	48.4064
17	1	4	3	0	29.9172	48.4179
18	1	0	2	4	33.6744	50.2973
19	1	0	4	2	33.6920	50.3055
20	2	0	1	4	33.7341	50.3264

TABLE II
Scene 1—Pairwise TTA Likelihood Table

	Sensor B				
Sensor A	1	2	3	4	0
1	0.9575	0.0052	0.0041	0.0000	0.0333
2	0.0041	0.9567	0.0000	0.0041	0.0352
3	0.0051	0.0004	0.9568	0.0052	0.0326
4	0.0000	0.0052	0.0041	0.9574	0.0333
0	0.0333	0.0326	0.0351	0.0334	0.0000

TABLE III
Scene 2—Top 20 Bias-Association Hypotheses

Hyp	Assignment				$-\log L(\mathbf{b},\mathbf{j})$	$-\log L(\mathbf{j})$
1	1	2	3	4	19.2874	44.2384
2	1	0	3	4	28.2314	47.4431
3	1	2	0	4	28.2361	47.4453
4	1	2	3	0	28.2388	47.4470
5	0	2	3	4	28.2438	47.4491
6	1	4	3	2	29.2532	49.2220
7	3	2	1	4	29.3967	49.2927
8	2	1	3	4	30.3936	49.7926
9	1	2	4	3	30.3991	49.7941
10	1	0	3	2	30.6948	48.6756
11	0	2	1	4	30.8173	48.7358
12	0	1	3	4	31.5877	49.1219
13	1	2	0	3	31.6002	49.1274
14	1	2	4	0	33.1373	49.8961
15	2	0	3	4	33.1447	49.9002
16	3	2	0	4	33.2380	49.9460
17	1	4	3	0	33.2405	49.9478
18	0	1	3	2	34.0486	50.3533
19	0	2	1	3	34.1829	50.4186
20	0	2	3	1	34.2054	50.4304

TABLE IV
Scene 2—Pairwise TTA Likelihood Table

	Sensor B				
Sensor A	1	2	3	4	0
1	0.9336	0.0059	0.0078	0.0000	0.0527
2	0.0110	0.9358	0.0000	0.0082	0.0450
3	0.0158	0.0000	0.9369	0.0059	0.0414
4	0.0016	0.0168	0.0109	0.9326	0.0381
0	0.0380	0.0415	0.0444	0.0533	0.0000

ation pairing table can shed some additional light onto the situation allowing the user to properly associate all tracks as needed.

Track Scene 2: Biased But Easy

Tables III and IV are configured in the same manner as Tables I and II. This will continue to be true for Scenes 3 and 4 in the following.

This scene has more bias than the previous and therefore the GNPM and MTTA likelihood values are slightly closer to one another with respect to the top two hypotheses. This logically provides values along the diagonal of Table IV that remain rather high, but are slightly lower than in the previous scene. This reduction in track pairing likelihoods along the main diagonal should be intuitively clear as the increased relative bias in the optimal hypotheses causes the contribution of the relative bias term in the likelihood functions to decrease (hence, the negative log likelihoods will increase). It is also worth noting that, just as in Track Scene 1, hypotheses 2 through 5 resemble the best hypothesis, hypothesis 1, except that only three of the four sensor A tracks are assigned, while the other is unassigned. If the gate value g were increased, these hypotheses, as well as all other hypotheses with null assignments, would become less favorable.

Track Scene 3: Slightly Ambiguous

This scene contains little bias but is more ambiguous than the previous two due to the existence of a fifth sensor B track. The results reflect this increased ambiguity in that the difference between hypotheses 1 and 2 is very slight. This is a case wherein the system-user may wish to look closely at Table VI for more information on the pairwise association confidence values. Although lower than in the previous two scenes, the diagonal values for tracks 1, 2 and 3 remain relatively high. However, the ambiguous portion (sensor A track 4 associates with either sensor B track 4 or 5) is readily observed as the values are 0.5043 and 0.4681, respectively. If the system-user is required to make a call regarding whether or not to fuse tracks, he may use this discrepancy to select $\{(1,1),(2,2),(3,3),(4,4)\}$ as the association vector. However, it is certainly acceptable in the fusion world to suggest that the results are inconclusive due to too much ambiguity. Indeed, in this scene, a reasonable alternative would be to fuse sensor A and B tracks 1, 2, and 3, while postponing

TABLE V
Scene 3—Top 20 Bias-Association Hypotheses

Hyp	Assignment				$-\log L(\mathbf{b},\mathbf{j})$	$-\log L(\mathbf{j})$
1	1	2	3	4	20.5577	41.3645
2	1	2	3	5	21.1202	41.6245
3	1	2	4	5	23.0836	42.6061
4	1	5	3	4	24.0485	43.0883
5	3	2	4	5	26.1490	44.1384
6	1	4	3	5	26.2722	44.2006
7	2	5	3	4	27.1338	44.6312
8	4	2	3	5	27.5214	44.8247
9	3	2	1	4	27.8584	45.0145
10	1	2	4	3	28.2426	45.2068
11	3	2	1	5	28.4286	45.2784
12	1	5	3	2	28.7029	45.4161
13	2	4	3	5	29.3567	45.7431
14	1	2	5	4	29.3879	45.7586
15	1	2	3	0	29.7613	44.7182
16	2	1	3	4	29.8116	45.9925
17	0	2	3	4	30.1844	44.9294
18	1	0	3	4	30.1853	44.9300
19	1	2	0	4	30.1870	44.9307
20	2	1	3	5	30.3742	46.2526

TABLE VI
Scene 3—Pairwise TTA Likelihood Table

	Sensor B					
Sensor A	1	2	3	4	5	0
1	0.9118	0.0254	0.0405	0.0117	0.0000	0.0106
2	0.0065	0.8689	0.0000	0.0266	0.0874	0.0106
3	0.0172	0.0000	0.8283	0.1393	0.0046	0.0106
4	0.0000	0.0065	0.0080	0.5043	0.4681	0.0131
0	0.0646	0.0992	0.1232	0.3180	0.4399	0.0000

a decision on sensor A track 4 and sensor B tracks 4 and 5.

Track Scene 4: Ambiguous

This scene is the most ambiguous of the four, which is reflected in the likelihoods of Tables VII and VIII. Tracks 1, 2, and 3 continue to have a high confidence of being the correct association. To some, the results may still look favorable, but to others the associations may not be strong enough to make a call. We leave these types of decision to the system-user.

With four concrete examples in hand, we close this section by briefly mentioning two ad hoc methods for computing individual track pairing likelihoods that were used in preliminary computational experiments [23]. We believe these approaches are inferior, but not altogether useless. First, it seems reasonable that one can get a general sense of how likely a particular track pairing is by simply counting the number of times sensor A track i is paired with sensor B track j among the top K hypotheses. For example, in the first track scene, sensor A track 1 is assigned to sensor B track 1 twelve times, to sensor B track 2 three times, to the dummy track three times, and to sensor B track 3 two times. Since sensor A track 1 and B track 1 are paired the most frequently, one might be willing to conclude that their data should be fused. Unfortunately, this approach quickly breaks down when considering the fourth track scene. In this scene, the most ambiguous of the four, we see that sensor A track 2 is assigned to sensor B tracks 2 and 5 ten times each. The "tallying" method is inadequate in this setting.

A second option is to augment the "tallying" method by giving each hypothesis an additional weight (as opposed to the uniform weight $1/K$ above). Rather than just count the number of times sensor A track i is paired with sensor B track j, one could choose

TABLE VII
Scene 4—Top 20 Bias-Association Hypotheses

Hyp	Assignment				$-\log L(\mathbf{b},\mathbf{j})$	$-\log L(\mathbf{j})$
1	1	2	3	4	19.6996	38.2521
2	1	2	3	7	21.5482	39.1769
3	1	2	6	4	21.7231	39.2636
4	1	5	3	4	21.8086	39.3065
5	1	2	3	5	21.9579	39.3816
6	1	2	3	6	22.3714	39.5884
7	1	5	6	7	22.4929	39.6492
8	1	2	6	7	22.5048	39.6553
9	1	5	3	7	22.5512	39.6784
10	1	5	6	4	22.8158	39.8099
11	1	2	4	7	23.2260	40.0157
12	1	5	4	7	23.4778	40.1416
13	1	2	6	5	23.5040	40.1546
14	1	2	4	5	23.6384	40.2217
15	2	5	6	7	23.9242	40.3653
16	2	5	4	7	24.1012	40.4535
17	3	5	6	7	24.1288	40.4670
18	1	5	3	6	24.2284	40.5168
19	4	5	6	7	24.4117	40.6084
20	3	2	6	4	24.5709	40.6874

TABLE VIII
Scene 4—Pairwise TTA Likelihood Table

	Sensor B							
Sensor A	1	2	3	4	5	6	7	0
1	0.8928	0.0475	0.0403	0.0194	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.6438	0.0000	0.0000	0.3562	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.5485	0.1174	0.0000	0.3340	0.0000	0.0000
4	0.0000	0.0000	0.0000	0.4122	0.1255	0.0752	0.3871	0.0000
0	0.1072	0.3087	0.4111	0.4509	0.5184	0.5908	0.6129	0.0000

to weight the hypotheses in some logical fashion. For example, hypothesis 1 out of 20 could get a weight of 20, hypothesis 2 a weight of 19,…, hypothesis 20 a weight of 1. As another example, one could use the GNPM likelihoods as weights. Finally, one could sum the total weights of the hypotheses in which tracks i and j are paired and normalize. Empirically, we have seen that employing a non-uniform weighting scheme often works reasonably well, and so the next logical question is: What is the correct weighting scheme? Indeed, in this paper, we argue that the MTTA likelihood values provide the "best" weights for the hypotheses. Moreover, while the ad hoc approaches in this second option may work, they lack probabilistic rigor. The MTTA likelihood has a solid mathematical backbone and, as these examples illustrate, furnishes insightful information for the system-user.

In summary, the above example illustrates the benefits of providing the system-user with several representations of the information that will be needed to assess the current situation and act appropriately. Although somewhat contrived, these vignettes clearly convey how both sensor bias and ambiguity could affect the results in a realistic tracking problem setting. We present both the GNPM and MTTA likelihood functions as measures for confidence in track-to-track association with sensor bias as well as illustrate how results will differ based on which one is chosen.

5. OPTIMIZING WITH RESPECT TO THE MTTA LIKELIHOOD

Equipped with a marginal 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 (8), as a mathematical program, specifically, as a 0-1 nonlinear optimization problem.

Starting from Equation (8), we collect the $(2\pi)^{D/2}$ terms and use the fact that $|I| + |I^0| = n_A$ to write

$$L(\mathbf{j}) = (\beta_{NTA}\beta_{NTB})^{n_A}(e^{(1/2)g})^{|I|}\frac{\sqrt{|\mathbf{V}|}}{\sqrt{\prod_{i\in I^+}|\mathbf{S}_{ij_i}|}}e^{-(1/2)\zeta}$$

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

$$-2\log L(\mathbf{j}) = -2\log(\beta_{NTA}\beta_{NTB})^{n_A} - |I|g - \log(|\mathbf{V}|) + \sum_{i\in I^+}\log(|\mathbf{S}_{ij_i}|) + \zeta.$$

Replacing $\mathbf{S}_{0j_0}$ with $\mathbf{R}$ and ζ with $(\sum_{i\in I}\mathbf{x}_{ij_i}^T\mathbf{S}_{ij_i}^{-1}\mathbf{x}_{ij_i}) - \mathbf{u}^T\mathbf{V}\mathbf{u}$, where $\mathbf{V} = (\mathbf{R}^{-1} + \sum_{i\in I}\mathbf{S}_{ij_i}^{-1})^{-1}$, we obtain

$$-2\log L(\mathbf{j}) = \kappa + |I^0|g - \log(|\mathbf{V}|) + \sum_{i\in I}(\mathbf{x}_i^T\mathbf{S}_{ij_i}^{-1}\mathbf{x}_{ij_i} + \log(|\mathbf{S}_{ij_i}|)) - \mathbf{u}^T\mathbf{V}\mathbf{u} \tag{10}$$

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

As in the GNPM problem, we 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, as well as the set $\mathcal{Y}$ of all feasible associations. Finally, since κ 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 0-1 optimization problem, which we will henceforth refer to simply as the *Marginal Track-to-Track Association problem* or MTTA problem for short:

$$\begin{aligned}\min_{\mathbf{y}} \quad & \mathbf{c}^T\mathbf{y} - \log(|\mathbf{V}(\mathbf{y})|) - (\mathbf{A}\mathbf{y})^T\mathbf{V}(\mathbf{y})(\mathbf{A}\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{V}(\mathbf{y}) = \left(\mathbf{R}^{-1} + \sum_{i=1}^{n_A}\sum_{j=1}^{n_B}\mathbf{S}_{ij}^{-1}y_{ij}\right)^{-1}, \qquad \mathbf{y}\in\mathcal{Y}\end{aligned} \tag{11}$$

where $\mathbf{A}$ is a $D \times n_A(n_B + 1)$ matrix of reals whose columns are such that

$$\mathbf{A}\mathbf{y} = \sum_{i=1}^{n_A}\sum_{j=0}^{n_B}\left\{\begin{array}{ll}\mathbf{S}_{ij}^{-1}(\mathbf{x}_i^A - \mathbf{x}_j^B) & \text{if} \quad j > 0 \\ \mathbf{0} & \text{if} \quad j = 0\end{array}\right\}y_{ij}$$

the coefficients

$$c_{ij} = c_{ij}(\mathbf{0}) = \begin{cases} d_{ij}^2(\mathbf{0}) + \log|\mathbf{S}_{ij}| & \text{if} \quad j \in N_B \\ g & \text{if} \quad j = 0\end{cases}$$

for $i \in N_A$, are given, and g is the gate value defined in Equation (3). The mathematical program (11) can be classified as a nonconvex nonlinear matching problem, which is a class of optimization problems that is generally difficult to solve to provable optimality. In contrast with the GNPM problem (4), which includes a continuous decision variable (vector) to model the relative bias that we hope to determine and, consequently, is formulated as a MINLP, the MTTA problem does not include any continuous decision variables.

6. SOLUTION METHODS

In this section, we discuss solution techniques for solving the MTTA 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 to provable optimality. Instead, highly-tailored algorithms that exploit idiosyncrasies of the particular problem are required. The MTTA 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 [23], which often arise given realistic data, the GNPM problem discussed in Section 2 provides a close approximation to the MTTA 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 multistart local search heuristics described in [5] and [25]. Essentially, these heuristics apply a fast iterative local search method from various starting points (hence the term "multistart") and find good bias-association hypotheses, $(\mathbf{b},\mathbf{j})$ pairs, to the GNPM problem. To recover the "pure" association likelihood of association hypothesis $\mathbf{j}$, one can solve Equation (8). Because these are heuristics, no optimality guarantee is provided. Nevertheless, these methods are extremely fast and have proven to be important when faced with real-time processing constraints.

Given that heuristics are already available for finding good association hypotheses, we turn to the question of solving the MTTA problem *exactly*, i.e., identifying a provably optimal solution or the K provably best solutions. 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 [18, 26], 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.

Below, we describe two different approaches for solving the MTTA problem to provable optimality. In the first approach, we solve the MTTA problem using a branch-and-bound method in which at each node in the search tree we solve (not necessarily to provable optimality) a quadratic assignment problem relaxation of the problem. As this approach does not scale well as the number of tracks increases, we provide a second approach in which we reformulate the MTTA problem as a mixed-integer linear optimization problem, which has many more variables and many more constraints than the original MTTA formulation, but performs remarkably well for a much larger set of instances of practical interest.

In what follows, we will need the following notation:

- The partial orderings $\mathbf{M} \succ \mathbf{0}$ and $\mathbf{M} \succeq \mathbf{0}$ mean that a square matrix $\mathbf{M}$ is positive definite and positive semidefinite, respectively. Given another square matrix $\mathbf{N}$, $\mathbf{M} \succ \mathbf{N}$ and $\mathbf{M} \succeq \mathbf{N}$ mean that $\mathbf{M} - \mathbf{N} \succ \mathbf{0}$ and $\mathbf{M} - \mathbf{N} \succeq \mathbf{0}$, respectively.
- We write $\mathbf{G} \cdot \mathbf{H} = \sum_i \sum_j g_{ij} h_{ij}$ to denote the Frobenius inner product of matrices $\mathbf{G}$ and $\mathbf{H}$.

6.1. Approach 1: A Branch-and-Bound Framework with Quadratic Assignment Problem Relaxations

In this section, we describe a branch-and-bound method that uses bounds obtained from solving a variant of the quadratic assignment problem at each node in the search tree. Branch-and-bound methods have been developed for the GNPM problem [5, 25] and closely resemble the one described below. The key difference between these methods and the one below is the construction of a lower bound at each node.

A systematic way of finding an optimal solution or the K best solutions to the MTTA 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., [22]). 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.

We now describe a non-traditional branching strategy that has worked well in practice in which a parent node may have more than two children.[3] 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

[3] In most commercial and non-commercial mixed-integer programming solvers, a parent node gives rise to at most two child nodes so that a so-called binary tree is maintained. Empirically, we have found that the branching scheme described here works very well for the GNPM and MTTA problems.

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 $j_2 \in \{0,1,\ldots,n_B\}\backslash\{j_1\}$. 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.

In order to compute a valid lower bound at each node in the search tree to be used for pruning, we solve (not necessarily optimally) a variant of the quadratic assignment problem (QAP). The QAP is one of the most studied and notoriously difficult optimization problems. For more details, we refer the reader to [4] and [29]. When stated as a facility location problem, the classical QAP is to assign n facilities to n locations such that the total interaction cost of all possible flow-distance products between the locations to which the facilities are assigned and the allocation costs of facilities to locations are minimized [29]. If we let $N = \{1,\ldots,n\}$ be a set of nodes, $\mathcal{A} = \{(i,j) \in N \times N\}$ be a set of arcs, then the Koopmans-Beckmann form of the classical QAP is

$$\min \quad \sum_{(i,j)\in\mathcal{A}} \sum_{(k,l)\in\mathcal{A}} d_{ijkl} x_{ij} x_{kl} - \sum_{(i,j)\in\mathcal{A}} f_{ij} x_{ij} \tag{12a}$$

$$\text{s.t.} \quad \sum_{j\in N} x_{ij} = 1, \qquad \forall i \in N \tag{12b}$$

$$\sum_{i\in N} x_{ij} = 1, \qquad \forall j \in N \tag{12c}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{A} \tag{12d}$$

where x_{ij} are assignment variables analogous to the y_{ij} variables in the MTTA problem, and d_{ijkl} and f_{ij} are data. In the generalized QAP, the number of facilities and locations need not be the same. In addition, the equality constraints (12c) are replaced with the more general inequality constraints $\sum_{i\in M} a_{ij} x_{ij} \le b_j$, $\forall j \in N$, where a_{ij} and b_j are data. Thus, constraints (12b) and (12c) are replaced by

$$\sum_{j\in N} x_{ij} = 1, \qquad \forall i \in M$$

$$\sum_{i\in M} a_{ij} x_{ij} \le b_j, \qquad \forall j \in N$$

where M is the set of locations and N is the set of facilities. The constraint set $\mathcal{Y}$ for the MTTA problem is therefore a special case of that of the generalized QAP in which a_{ij} and b_j are 1 for all $i \in M$ and $j \in N$.

6.1.1. Deriving Lower Bounds via QAP Relaxations

The basic idea behind obtaining a QAP relaxation of the MTTA problem at each node is as follows: At each depth $d \in \{0,1,\ldots,n_A-1\}$ in the search tree, we replace the matrix $\mathbf{V}(\mathbf{y})$, which depends on the decision vector $\mathbf{y}$ and consequently introduces unwieldy nonlinearities, with a suitable chosen positive definite matrix $\mathbf{V}_d$ such that

$$\mathbf{V}_d^{-1} := \mathbf{R}^{-1} + \sum_{i=1}^{d} \sum_{j=1}^{n_B} \mathbf{S}_{ij}^{-1} y_{ij}.$$

This leads to the following special case of the Generalized QAP at each node:

$$z^{\text{QAP}} = -\log(|\mathbf{V}_d|) + \min \ \mathbf{c}^T\mathbf{y} - \mathbf{y}^T\mathbf{Q}_d\mathbf{y} \tag{13a}$$

$$\text{s.t.} \quad \mathbf{y} \in \mathcal{Y} \tag{13b}$$

where $\mathbf{Q}_d = \mathbf{A}^T\mathbf{V}_d\mathbf{A}$. Notice that after replacing the matrix $\mathbf{V}(\mathbf{y})$ with a fixed matrix $\mathbf{V}_d$, the term $-\log(|\mathbf{V}_d|)$ becomes a constant and can be removed from the minimization. In what follows, we refer to the minimization problem in (13) as NodeQAP. In addition, note that NodeQAP is a special case of the generalized QAP stated as a maximization problem rather than as a minimization problem, which is more typical.

We now justify that this approach of replacing the matrix $\mathbf{V}(\mathbf{y})$ with the matrix $\mathbf{V}_d$ and solving NodeQAP provides a valid lower bound at each node in the search tree. The following proposition leads directly to the justification.

PROPOSITION 1 *For all depths $d \in \{0,1,\ldots,n_A-1\}$, define $\mathbf{V}_d^{-1} := \mathbf{R}^{-1} + \sum_{i=1}^{d}\sum_{j=1}^{n_B} \mathbf{S}_{ij}^{-1} y_{ij}$. Then,*

$$\mathbf{V}_{d+1}^{-1} \succeq \mathbf{V}_d^{-1} \qquad \textit{and} \qquad -\log|\mathbf{V}_{d+1}| \ge -\log|\mathbf{V}_d|.$$

PROOF Since the matrices $\mathbf{S}_{ij}$ and $\mathbf{S}_{ij}^{-1}$ are positive definite for all $i \in N_A$ and $j \in N_B$, it follows that $\mathbf{V}_{d+1}^{-1} = \mathbf{V}_d^{-1} + \mathbf{S}_{d+1,j}^{-1} y_{d+1,j} \succeq \mathbf{V}_d^{-1}$ for $d = 0,\ldots,n_A-1$ and for all $j \in \{0,\ldots,n_B\}$. Moreover,

$$\begin{aligned}
\mathbf{V}_{d+1}^{-1} \succeq \mathbf{V}_d^{-1} &\iff \mathbf{V}_{d+1} \preceq \mathbf{V}_d \\
&\iff |\mathbf{V}_{d+1}| \le |\mathbf{V}_d| \\
&\iff \log|\mathbf{V}_{d+1}| \le \log|\mathbf{V}_d| \\
&\iff -\log|\mathbf{V}_{d+1}| \ge -\log|\mathbf{V}_d|
\end{aligned}$$

where the first equivalence is a linear algebra exercise and 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.

In words, the proposition tells us that the matrix $\mathbf{V}_{d+1}^{-1}$ associated with a child node is at least as positive definite as that of its parent, and can only become more positive definite than that of its parent if sensor A track $d+1$ is assigned to a sensor B track $j \in N_B$. The implication of the proposition is: Since

$$\mathbf{y}^T\mathbf{A}^T\mathbf{V}_d\mathbf{A}\mathbf{y} \geq \mathbf{y}^T\mathbf{A}^T\mathbf{V}_{d+1}\mathbf{A}\mathbf{y}, \qquad \forall \mathbf{y} \in \mathcal{Y}\ ^4$$

and $-\log|\mathbf{V}_{d+1}| \geq -\log|\mathbf{V}_d|$, the optimal value z^{QAP} of (13) at a parent node at depth d will always be no greater than the optimal value at one of its descendants at depth $d+1, d+2, \ldots, n_A$. Therefore, z^{QAP} is a valid lower bound for the optimal objective function value of a node at depth d in the search tree.

6.1.2. Solving NodeQAP

There are several issues to consider when solving NodeQAP: How much time should be spent computing a lower bound at each node? Which formulation to use? Which algorithm to use?

We start with the first question. Our main goal of using a QAP relaxation at each node is to obtain as quickly as possible a good lower bound to facilitate pruning. Thus, it is not necessary to solve NodeQAP to optimality at every node since we simply need to determine whether or not this node has the potential to lead to one of the K best solutions. One is then faced with the following trade-off:

- Option 1: Spend a small amount of time at each node so that many nodes can be processed quickly, while possibly sacrificing our ability to prune nodes early in the tree;
- Option 2: Spend more time computing the best possible lower bound so that fewer nodes need to be expanded.

Since the classical QAP is a very difficult problem in its own right, empirically we have found it best to obtain a lower bound on NodeQAP at nodes with a small depth while forgoing optimality. Specifically, in our computational experiments, we set a time limit of a fraction of a second to obtain a lower bound at each node. We observed that even solving NodeQAP to provable optimality at a small depth (e.g., $d \leq n_A/4$) in the search tree rarely allows us to prune nodes. Consequently, it makes sense not to waste computational effort computing a bound that will not be immediately useful. On the other hand, as we dive deeper in the search tree, the lower bound furnished by (13) becomes stronger and more useful, while also becoming easier to solve to optimality since more assignments are fixed.

Regarding the question of which formulation to use, we have experimented with two approaches: reformulating NodeQAP as a mixed-integer linear program and relaxing NodeQAP as a semidefinite program. Semidefinite programming approaches are discussed in [4] and our initial semidefinite programming formulation is discussed in [24]; we will not discuss this approach here since, despite extensive experimentation, semidefinite programming software is not yet as mature as that for mixed-integer programming.

In our computations, we use the Kaufman-Broeckx linearization (1.12) on p.9 of [4], which is arguably the smallest linearization in terms of the number of variables and constraints in the model. The Kaufman-Broeckx formulation is a mixed-integer programming formulation of the QAP, which we solve using the commercially-available solver CPLEX 11.2 [13]. As discussed above, at shallow depths in the search tree, we have CPLEX compute the best lower bound for the Kaufman-Broeckx formulation of NodeQAP that it can provide in a fraction of a second (typically, by solving the linear programming relaxation of the Kaufman-Broeckx formulation of NodeQAP and performing a partial branch-and-bound). Later, at deeper depths in the search tree, CPLEX can solve the Kaufman-Broeckx formulation of NodeQAP to provable optimality in under a second.

Another important algorithmic enhancement is to recognize that it is not necessary to solve NodeQAP from scratch at each node. Indeed, information from the solution at a parent node should be exploited to help solve the QAP at a child node. In particular, there are two items that change from a parent node at depth d to a child node at depth $d+1$: the matrix $\mathbf{A}^T\mathbf{V}_d\mathbf{A}$ of the quadratic term in the objective function becomes $\mathbf{A}^T\mathbf{V}_{d+1}\mathbf{A}$ and an additional assignment constraint $y_{d,j_d} = 1$ appears in the child node QAP. At every node, we store the best lower bound and the best feasible solution computed in the allotted time limit. We then use this information to warm-start the solution procedure at each child node. Most mixed-integer programming solvers have options for warm-starting an algorithm using one or more known feasible solutions (see, e.g., [13]).[5]

6.1.3. A Branch-and-Bound Algorithm

We conclude this subsection with a high-level description of our branch-and-bound algorithm and with a proof of its correctness. Pseudocode is provided in Algorithm 1. Let $z(\mathbf{y}) = \mathbf{c}^T\mathbf{y} - \log|\mathbf{V}(\mathbf{y})| - (\mathbf{A}\mathbf{y})^T\mathbf{V}(\mathbf{y})(\mathbf{A}\mathbf{y})$. Let $z^{\mathrm{QAP}}(\mathtt{node})$ be the value of z^{QAP} in (13) at a particular node, call it `node`, in the search tree. Recall that at a

[4] Actually, this inequality holds for all $\mathbf{y} \in \mathbb{R}^{n_A(n_B+1)}$.

[5] In fact, it is precisely our ability to re-optimize NodeQAP at a child node given the solution at its parent that makes the mixed-integer programming formulation more attractive than the semidefinite programming formulation. Current semidefinite programming software does not have this algorithmic enhancement available.

node at depth $d \in \{0,1,\ldots,n_A\}$ in the search tree, the assignments of first d sensor A tracks are fixed. As is typical in a branch-and-bound implementation, we maintain a priority queue `pq` of nodes, whose `push(node)` method places `node` on the queue and whose `pop()` method (sometimes called a `findMin()` method) returns and removes the node with a minimum $z^{\text{QAP}}(\texttt{node})$ value from the queue. Note that this is a *best-first search* implementation and that the $z^{\text{QAP}}(\texttt{node})$ value of the node returned by the `pop()` method always provides a lower bound on the objective function value of the Kth provably best solution. Finally, the algorithm makes use of another method `createBranches(node,pq)`, which takes a given non-leaf node, creates its children as described in the beginning of this subsection, and adds these descendant nodes to the priority queue.

ALGORITHM 1 `BranchAndBoundforMTTA`(K)

1: Find K high-quality incumbent solutions $\mathbf{y}^1,\ldots,\mathbf{y}^K$ with $z(\mathbf{y}^1) \le \cdots \le z(\mathbf{y}^K)$
2: Initialize a priority queue of nodes: `pq={}`
3: Set $z^{UB} = z(\mathbf{y}^K)$; `pq.push(rootnode)`
4: **while** `pq` is not empty **do**
5: `node=pq.pop()`
6: **if** `node` is a leaf node **then**
7: Let $\hat{\mathbf{y}}$ be the solution associated with `node`
8: **if** $z(\hat{\mathbf{y}}) < z^{UB}$ and $\hat{\mathbf{y}} \notin \{\mathbf{y}^1,\ldots,\mathbf{y}^K\}$ **then**
9: Update list of K best solutions to include $\hat{\mathbf{y}}$
10: Update z^{UB}: $z^{UB} = z(\mathbf{y}^K)$
11: **else**
12: Prune `node`
13: **end if**
14: **else if** $z^{\text{QAP}}(\texttt{node}) > z^{UB}$ **then**
15: Prune `node`
16: **else**
17: `createBranches(node,pq)`
18: **end if**
19: **end while**
20: **return** K best track-to-track association solutions

THEOREM 1 (Algorithm 1 is exact) *The proposed branch-and-bound procedure finds the K provably best (globally optimal) solutions to the MTTA problem* (11).

PROOF The proof is by contradiction. Suppose Algorithm 1 returns the solutions $\mathbf{y}^1,\ldots,\mathbf{y}^K$ with $z(\mathbf{y}^1) \le \cdots \le z(\mathbf{y}^K)$. Suppose, to arrive at a contradiction, that there exists some solution $\hat{\mathbf{y}} \ne \mathbf{y}^k$, for $k = 1,\ldots,K$, such that $z(\hat{\mathbf{y}}) < z(\mathbf{y}^K)$. There are only two possible ways for (the leaf node corresponding to) the solution $\hat{\mathbf{y}}$ to have been pruned: implicitly or explicitly. If $\hat{\mathbf{y}}$ was pruned implicitly, then one of its ancestors in the search tree, call this node `ancestor`, had to be pruned in Step 15 because $z^{\text{QAP}}(\texttt{ancestor}) > z^{UB}$. Note that $z^{UB} \ge z(\mathbf{y}^K)$ since z^{UB} is an upper bound on the objective function value of the Kth best solution. But this immediately implies a contradiction since

$$z^{\text{QAP}}(\texttt{ancestor}) \le z(\hat{\mathbf{y}}) < z(\mathbf{y}^K) \le z^{UB} < z^{\text{QAP}}(\texttt{ancestor})$$

where the first inequality follows from Proposition 1 and the discussion that followed. On the other hand, if $\hat{\mathbf{y}}$ was pruned explicitly, then $\hat{\mathbf{y}}$ was pruned in Step 12. But this too implies a contradiction since, by assumption, $z(\hat{\mathbf{y}}) < z(\mathbf{y}^K)$. Hence, Algorithm 1 never incorrectly prunes a node, and therefore finds the K certifiably best solutions to the MTTA problem.

6.2. Approach 2: A Branch-and-Cut Algorithm for a MIP Reformulation

For our second approach, we reformulate the MTTA problem as a mixed-integer linear program (MIP) and then solve it using a branch-and-cut algorithm with a commercially-available MIP solver. A branch-and-cut algorithm is a more advanced branch-and-bound algorithm in which valid inequalities, known as "cutting planes" or simply as "cuts," are appended to the original formulation at each node in order to strengthen the formulation, improve bounds, and expedite node pruning. Virtually all commercial MIP solvers use a branch-and-cut algorithm rather than a pure branch-and-bound algorithm (see, for example, [13]).

Arriving at our proposed MIP reformulation takes two steps. In the first step, we introduce additional continuous auxiliary decision variables to reformulate the MTTA problem as a MINLP which possesses a special structure. Namely, the objective function consists of a submodular set function and a linear function, while the feasible region is a mixed-integer linear set, i.e., a set comprised of continuous and integer decision variables and linear inequalities. In other words, the only nonlinearity appears in the form of a submodular set function. In the second step, using well-known techniques for submodular function minimization, we replace the submodular function with a continuous auxiliary decision variable bounded below by a finite number of linear inequalities. This substitution gives rise to the desired MIP formulation, albeit one with a factorial number (in the number of tracks) of inequalities. Finally, we show how to generate only those cuts that are necessary in the formulation on an "as-needed" basis. Empirically, we have found this approach to be far superior to the branch-and-bound approach described above.

6.2.1. Towards a MINLP Reformulation

As outlined above, our first step towards achieving the desired MIP model is to reformulate the MTTA problem as a MINLP, which is a mixed-integer linear model except for a submodular function appearing in the objective function. To this end, note that the term $\mathbf{y}^T\mathbf{A}^T\mathbf{V}(\mathbf{y})\mathbf{A}\mathbf{y}$ in the objective function of (11) appears to be quadratic in $\mathbf{y}$, but is not because the matrix $\mathbf{V}(\mathbf{y})$ is the *inverse* of a sum of positive definite matrices

which depend on the decision variable $\mathbf{y}$. Our first goal is to circumvent the need to take an inverse by linearizing this term into a form suitable for a MIP solver. Consequently, we introduce an auxiliary decision vector $\mathbf{z} = \mathbf{V}(\mathbf{y})\mathbf{A}\mathbf{y}$ such that $\mathbf{V}^{-1}(\mathbf{y})\mathbf{z} = \mathbf{A}\mathbf{y}$. Recall that the inverse exists since all matrices are assumed to be positive definite. Then,

$$\begin{aligned}\mathbf{A}\mathbf{y} &= \mathbf{R}^{-1}\mathbf{z} + \sum_{i=1}^{n_A}\sum_{j=1}^{n_B}\mathbf{S}_{ij}^{-1}\mathbf{z}y_{ij} \\ &= \mathbf{R}^{-1}\mathbf{z} + \sum_{i=1}^{n_A}\sum_{j=1}^{n_B}\mathbf{S}_{ij}^{-1}\mathbf{w}_{ij}\end{aligned}$$

where $\mathbf{w}_{ij}$ is a D-dimensional vector of auxiliary decision variables such that

$$\begin{aligned}\mathbf{w}_{ij} &= \mathbf{z}y_{ij} \\ &= \begin{cases}\mathbf{z} & \text{if} \quad y_{ij} = 1, \\ \mathbf{0} & \text{if} \quad y_{ij} = 0,\end{cases} \qquad \forall i \in N_A, \quad j \in N_B \cup \{0\}\end{aligned}$$

and $\mathbf{w}_{i0} = \mathbf{0}$, $\forall i \in N_A$. Furthermore, we can linearize the nonlinear equations $\mathbf{w}_{ij} = \mathbf{z}y_{ij}$ (recall that both $\mathbf{z}$ and y_{ij} are decision variables) by replacing them with the following linear constraints:

$$-My_{ij}\mathbf{1} \le \mathbf{w}_{ij} \le My_{ij}\mathbf{1} \tag{14a}$$

$$-M(1-y_{ij})\mathbf{1} \le \mathbf{z} - \mathbf{w}_{ij} \le M(1-y_{ij})\mathbf{1} \tag{14b}$$

for all $i \in N_A$ and $j \in N_B$, where M is an appropriately chosen parameter and $\mathbf{1}$ is a D-dimensional vector of ones. This type of linearization of bi-linear terms is a standard "trick" in discrete optimization. Moreover, the nonlinear term $\mathbf{y}^T\mathbf{A}^T\mathbf{V}(\mathbf{y})\mathbf{A}\mathbf{y}$ in the objective function becomes $\mathbf{A}\cdot\mathbf{W}$, where $\mathbf{W} \in \mathbb{R}^{D\times n_A\times(n_B+1)}$ is the matrix satisfying $\mathbf{A}\cdot\mathbf{W} = \mathbf{y}^T\mathbf{A}^T\mathbf{z}$. This done, we have arrived at the following MINLP reformulation of the MTTA problem:

$$\min \quad f(\mathbf{y}) + \mathbf{c}^T\mathbf{y} - \mathbf{A}\cdot\mathbf{W} \tag{15a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{y} = \mathbf{R}^{-1}\mathbf{z} + \sum_{i=1}^{n_A}\sum_{j=1}^{n_B}\mathbf{S}_{ij}^{-1}\mathbf{w}_{ij} \tag{15b}$$

$$\text{(14a), (14b)}, \quad \mathbf{w}_{i0} = \mathbf{0}, \quad \forall i \in N_A, \quad \forall j \in N_B \tag{15c}$$

$$\mathbf{w}_{ij} \in \mathbb{R}^D, \quad \forall i \in N_A, \quad \forall j \in N_B \cup \{0\} \tag{15d}$$

$$\mathbf{y} \in \mathcal{Y}, \quad \mathbf{z} \in \mathbb{R}^D \tag{15e}$$

where $f(\mathbf{y}) := -\log(|\mathbf{V}(\mathbf{y})|) = \log(|\mathbf{V}^{-1}(\mathbf{y})|) = \log(|\mathbf{R}^{-1} + \sum_{i=1}^{n_A}\sum_{j=1}^{n_B}\mathbf{S}_{ij}^{-1}y_{ij}|)$. To reiterate, with the exception of the nonlinear function $f(\mathbf{y})$ in the objective function, the above formulation is a MIP.

6.2.2. Towards a MIP Reformulation

The second step in obtaining the desired MIP reformulation is to show that the function $f(\mathbf{y})$ is a submodular set function and then describe how to exploit this submodularity in a branch-and-cut algorithm. Submodular functions have undergone extensive study in the optimization and computer science communities. We refer the reader to [1] and the references therein.

In what follows, let $n = n_A(n_B + 1)$ and define the set $N := \{1,\dots,n\}$. We also follow the common practice in the submodular function literature of abusing notation so that we may refer to a set function $h(S)$, for some $S \subseteq N$, as $h(\mathbf{x})$, where $\mathbf{x} \in \{0,1\}^n$ is the indicator vector of subsets of N. In addition, let $\mathbf{x}(S)$ denote the indicator vector of a set $S \subseteq N$, i.e., $x_i(S) = 1$ if $i \in S$, and 0 otherwise, for each component $i = 1,\dots,n$. Let $S_{\mathbf{x}}$ denote the support of a vector $\mathbf{x}$, i.e., $i \in S_{\mathbf{x}}$ if $x_i = 1$ and $i \notin S_{\mathbf{x}}$ if $x_i = 0$.

DEFINITION 1 A set function $h : 2^N \mapsto \mathbb{R}$ is *submodular* on N if

$$h(S) + h(T) \ge h(S\cup T) + h(S\cap T), \qquad \forall S,T \subseteq N.$$

A submodular function can also be characterized by its difference function $\rho_k(S) := h(S\cup\{k\}) - h(S)$ for $S \subseteq N$ and $k \in N\backslash S$. It is easy to verify that a set function h is submodular if and only if its difference function is nonincreasing; that is, $\rho_k(S) \ge \rho_k(T)$, $\forall S \subseteq T \subseteq N$ and all $k \in N\backslash T$. The following proposition shows that the function $f(\mathbf{y})$ is submodular on N by validating that the difference function of h is nonincreasing.

PROPOSITION 2 *Let* $N = \{1,\dots,n\}$ *and suppose that* $\mathbf{A}_i \succ \mathbf{0}$, $\forall i \in N \cup \{0\}$. *Then, the function* $g(S) := \log(|\mathbf{A}_0 + \sum_{j\in S}\mathbf{A}_j|)$ *is submodular on* N.

PROOF Let $S \subseteq T \subseteq N$ and $k \in N\backslash T$. Let $\mathbf{B} = \mathbf{A}_0 + \sum_{j\in S}\mathbf{A}_j$, $\mathbf{C} = \mathbf{A}_0 + \sum_{j\in T}\mathbf{A}_j$, and note that $\mathbf{C} \succ \mathbf{B}$ and $\mathbf{B}^{-1} \succ \mathbf{C}^{-1}$. Then,

$$\begin{aligned}g(S\cup\{k\}) - g(S) &= \log(|\mathbf{B} + \mathbf{A}_k|) - \log(|\mathbf{B}|) \\ &= \log\left(\frac{|\mathbf{B} + \mathbf{A}_k|}{|\mathbf{B}|}\right) \\ &= \log(|\mathbf{I} + \mathbf{A}_k\mathbf{B}^{-1}|) \\ &\ge \log(|\mathbf{I} + \mathbf{A}_k\mathbf{C}^{-1}|) \\ &= g(T\cup\{k\}) - g(T)\end{aligned}$$

where the inequality follows from the fact that $\mathbf{B}^{-1} \succ \mathbf{C}^{-1}$ and multiplying both sides by a positive definite matrix does not affect this ordering, nor does adding a positive definite matrix to both sides.

Since f satisfies the conditions of Proposition 2, f is submodular on N. We now present some well-known results from the theory of submodular function minimization, which will allow us to reformulate the MINLP as a MIP. The majority of the background material below is taken from Section 2 of Atamtürk and Narayanan [1]. Our basic goal is to replace the problem of minimizing a submodular set function $h(S)$ over all possible subsets S of N with an equivalent optimization problem of minimizing an auxiliary continuous variable

θ, representing $h(S)$, over the convex lower envelope of h, which is a polyhedron whose linear inequality description is already known to us due to the fact that h is submodular. With these linear inequalities in hand, we can transform our MINLP into a MIP, which can be solved by a MIP solver.

Suppose we are trying to minimize a submodular set function $h : 2^N \mapsto \mathbb{R}$ over the set N:

$$\min\{h(S) : S \subseteq N\}. \tag{16}$$

This is equivalent to

$$\min\{\theta : \theta \geq h(\mathbf{x}),\ \mathbf{x} \in \{0,1\}^n\}$$

or

$$\min\{\theta : (\theta, \mathbf{x}) \in \mathcal{Q}_h\} \tag{17}$$

where

$$\mathcal{Q}_h := \text{conv}\{(\theta, \mathbf{x}) \in \mathbb{R} \times \{0,1\}^n : \theta \geq h(\mathbf{x})\}$$

is the convex hull of the epigraph of h, which Atamtürk and Narayanan [1] refer to as the convex lower envelope of h; $\mathcal{Q}_h$ is also a polyhedron.[6] Without loss of generality, we may assume that $h(\emptyset) = 0$, since otherwise we can solve the equivalent minimization problem for the shifted function h' with $h'(S) := h(S) - h(\emptyset)$, $\forall S \subseteq N$.

It turns out that the linear inequalities describing $\mathcal{Q}_h$ are intimately related to the points in the *extended polymatroid* associated with h, which is defined as

$$EP_h := \{\pi \in \mathbb{R}^n : \pi(S) \leq h(S),\ \forall S \subseteq N\}$$

where $\pi(S) := \sum_{i \in S} \pi_i$. Atamtürk and Narayanan [1] formalize this relationship between the points in EP_h and the valid inequalities for $\mathcal{Q}_h$ in the following proposition.

PROPOSITION 3 (Atamtürk and Narayanan [1]) *The inequality $\pi^T \mathbf{x} \leq \theta$ is valid for $\mathcal{Q}_h$ (i.e., $\pi^T \mathbf{x} \leq \theta$, $\forall (\theta, \mathbf{x}) \in \mathcal{Q}_h$) if and only if $\pi \in EP_h$.*

PROOF Given $\pi \in EP_h$, we have $\pi^T \mathbf{x} = \pi(S_\mathbf{x}) \leq h(S_\mathbf{x}) \leq \theta$. Conversely, if $\pi \notin EP_h$, then $\pi(S) > h(S)$ for some $S \subseteq N$; but then for $\theta = h(S)$, $\pi(S) = \pi^T \mathbf{x}(S) > \theta$, contradicting the validity of $\pi^T \mathbf{x} \leq \theta$.

A fundamental goal in integer programming is to identify which are the important linear inequalities that are necessary in describing a polyhedron or a set of integer points and, therefore, provide the tightest formulation possible; these "important" inequalities are known as *facets* [22]. We call the bound constraints $\mathbf{0} \leq \mathbf{x} \leq \mathbf{1}$ *trivial* facets of $\mathcal{Q}_h$ since they can be shown to be facets of $\mathcal{Q}_h$ and are obtained trivially from relaxing the constraint $\mathbf{x} \in \{0,1\}^n$ to $\mathbf{x} \in [0,1]^n$. Although Proposition 3 allows us to determine whether or not a linear inequality is valid for $\mathcal{Q}_h$, it does not tell us if the inequality is a nontrivial facet of $\mathcal{Q}_h$. The next proposition allows us to identify nontrivial facets of $\mathcal{Q}_h$ and gives us an important relationship between the extreme points of the extended polymatroid and the convex hull of the epigraph. Since the proof given below was provided in an earlier version of [1], but does not appear in their published version, we state the proof for completeness.

PROPOSITION 4 *The inequality $\pi^T \mathbf{x} \leq \theta$ is a nontrivial facet for $\mathcal{Q}_h$ if and only if π is an extreme point of EP_h.*

PROOF $\Rightarrow$ From Proposition 3, if $\pi \notin EP_h$, inequality $\pi^T \mathbf{x} \leq \theta$ is not valid for $\mathcal{Q}_f$. If $\pi \in EP_h$ is not an extreme point of EP_h, then $\pi = \lambda \pi^1 + (1-\lambda)\pi^2$ for some $\lambda \in (0,1)$ and distinct points $\pi^1, \pi^2 \in EP_h$ and $\pi^T \mathbf{x} \leq \theta$ is implied by $(\pi^1)^T \mathbf{x} \leq \theta$ and $(\pi^2)^T \mathbf{x} \leq \theta$.

$\Leftarrow$ Conversely, if π is an extreme point of EP_h, it is the unique solution to a set of n linearly independent equations $\pi(S_i) = h(S_i)$ for $i = 1, \ldots, n$. Then, the corresponding linearly independent points $(\mathbf{x}(S_i), h(S_i))$, for $i = 1, \ldots, n$ of $\mathcal{Q}_h$ and the origin $(\mathbf{0}, 0)$ are on the face $\{(\mathbf{x}, \theta) \in \mathcal{Q}_h : \pi^T \mathbf{x} = \theta\}$. Finally, as $(\mathbf{0}, 1) \in \mathcal{Q}_h$, but is not on the face, the face is proper.

REMARK 1 Note that if $h(\emptyset) \neq 0$, the nontrivial facets for Q_h take the form $\pi^T \mathbf{x} \leq \theta - h(\emptyset)$, $\forall \pi \in EP_{h'}$, where $h' := h - h(\emptyset)$. Consequently, since $f(\emptyset) = \log(|\mathbf{R}|)$, the nontrivial facets of Q_f are of the form $\theta \geq \log(|\mathbf{R}|) + \pi^T \mathbf{y}$, $\forall \pi \in EP_f$.

The significance of Proposition 4 is that it tells us almost precisely which inequalities are needed to define the polyhedron $\mathcal{Q}_h$; the only piece of information that is missing is how to compute the coefficients π_k of these facets. Fortunately, this is provided in the following important result due to Edmonds [6].

THEOREM 2 (Edmonds [6]) *The point π is an extreme point of EP_h if and only if $\pi_i = h(S_{(i)}) - h(S_{(i-1)})$, where $S_{(i)} = \{(1), (2), \ldots, (i)\}$, for $i = 1, \ldots, n$, and $S_0 = \emptyset$, for some permutation $((1), (2), \ldots, (n))$ of N.*

In words, Proposition 4 and Theorem 2 state that there are $n!$ nontrivial facets needed to describe the polyhedron Q_h, one for each permutation of the elements of N, and the precise value of each coefficient depends on the corresponding permutation. When specialized to the submodular function f, Proposition 4, Remark 1, and Theorem 2 lead immediately to the following corollary.

COROLLARY 1 *The $n!$ nontrivial facets of the polyhedron Q_f, which we refer to as extended polymatroid (EP) inequalities, are of the form*

$$\theta \geq \log(|\mathbf{R}|) + \sum_{k=1}^{n} \pi_k^\psi y_{\psi(k)}, \qquad \forall \psi \in \Psi_n \tag{18}$$

where Ψ_n is the set of all permutations of $\{1, \ldots, n\}$ and π_k^ψ is the kth coefficient in permutation $\psi = \{(1), \ldots, (n)\}$ of Ψ_n. More precisely, $\pi_k^\psi = f(S_{(k)}) - f(S_{(k-1)}) = \log(|\mathbf{R}^{-1} + \sum_{a=1}^{k} \mathbf{S}_{(a)}^{-1}|) - \log(|\mathbf{R}^{-1} + \sum_{a=1}^{k-1} \mathbf{S}_{(a)}^{-1}|)$, where a denotes the track pair (i, j) associated with the ath element of the permutation ψ.

[6] This fact is important as it assures us that $\mathcal{Q}_h$ can be described by a *finite* number of linear inequalities.

REMARK 2 If $\theta < f(\mathbf{y})$ and $\mathbf{y} \in [0,1]^n$, then, by definition, the point $(\theta, \mathbf{y}) \notin Q_f$ and hence, by Corollary 1, at least one EP inequality (18) is violated.[7]

Using the equivalence that allowed us to express (16) as (17), and by applying Corollary 1 to provide us with an explicit polyhedral representation of Q_f, we can transform Formulation (15) and pose the MTTA problem as the following MIP:

$$\min \quad \theta + \mathbf{c}^T\mathbf{y} - \mathbf{A}\cdot\mathbf{W} \tag{19a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{y} = \mathbf{R}^{-1}\mathbf{z} + \sum_{i=1}^{n_A}\sum_{j=1}^{n_B}\mathbf{S}_{ij}^{-1}\mathbf{w}_{ij} \tag{19b}$$

$$(14a),\ (14b), \quad \mathbf{w}_{i0} = \mathbf{0}, \quad \forall i \in N_A, \quad \forall j \in N_B \tag{19c}$$

$$\mathbf{w}_{ij} \in \mathbb{R}^D, \quad \forall i \in N_A, \quad \forall j \in N_B \tag{19d}$$

$$\mathbf{z} \in \mathbb{R}^D, \quad \mathbf{y} \in \mathcal{Y}, \quad \theta \geq \log(|\mathbf{R}|) \tag{19e}$$

$$\theta \geq \log(|\mathbf{R}|) + \sum_{k=1}^{n}\pi_k^{\psi} y_{\psi(k)}, \quad \forall \psi \in \Psi_n. \tag{19f}$$

In contrast to the original formulation of the MTTA problem (11) and the formulation of NodeQAP (13), both of which operate in the space ($\mathbb{R}^n$) of the original decision variables $\mathbf{y}$, Formulation (19) requires us to introduce $O(Dn)$ auxiliary decision variables, thus forcing us to operate in a higher-dimensional space, and $O(3n)$ additional constraints prior to introducing constraints (19f). At first glance, such a transformation may seem "indirect" and fruitless. However, the advantage of this formulation is that it is a MIP and can, in theory, be solved by an off-the-shelf MIP solver. Indeed, the benefits of working in a higher-dimensional space will become clear in Section 7.

6.2.3. A Branch-and-Cut Algorithm

Unfortunately, the above MIP formulation requires a factorial number ($n!$) of EP inequalities (19f), which is unwieldy for instances of practical interest. However, it turns out that this drawback can easily be overcome. Since very few of these EP constraints will be tight at an optimal solution, rather than include them all in the initial formulation, we omit all of them from the outset, and then generate those that are necessary on an "as-needed" basis. Specifically, we initially formulate the MIP (19a)–(19e). Then, at every node in the search tree, we check if $\hat{\theta} \geq f(\hat{\mathbf{y}}) = \log(|\mathbf{R}^{-1} + \sum_{i=1}^{n_A}\sum_{j=1}^{n_B}\mathbf{S}_{ij}^{-1}\hat{y}_{ij}|)$ where $\hat{\theta}$ and $\hat{\mathbf{y}}$ are partial solutions (obtained from the linear programming relaxation) at that node. Note that $\hat{\mathbf{y}}$ will have fractional components at a non-leaf node in the search tree. If $\hat{\theta} < f(\hat{\mathbf{y}})$, then, by definition, $(\hat{\theta}, \hat{\mathbf{y}}) \notin Q_f$ and so we "cut off" the solution at that node by appending the most violated EP inequality, which is guaranteed to exist by Remark 2. Edmonds [6] showed that finding the most violated EP inequality can be solved very efficiently using "the Greedy Algorithm" 2.

ALGORITHM 2 `GreedyAlgorithm`$(f, \hat{\mathbf{y}})$

1: Sort the $\hat{y}_i$ variables in nonincreasing order, $\hat{y}_{[1]} \geq \cdots \geq \hat{y}_{[n]}$, breaking ties arbitrarily
2: Define $S_{[i]} := \{[1],[2],\ldots,[i]\}$, for $i = 1,\ldots,n$, and $S_0 := \emptyset$
3: Define $\pi_{[i]} := f(S_{[i]}) - f(S_{[i-1]})$, for $i = 1,\ldots,n$
4: **return** The EP inequality $\pi^T\mathbf{y} \leq \theta - f(\emptyset)$

A high-level sketch of the algorithm used to find the single provably best (optimal) solution to the MTTA problem (11) is outlined in Algorithm 3. The pseudocode is more terse than that given in Algorithm 1 because we have implemented our algorithm in a commercial solver (CPLEX [13]) which manages the branch-and-bound tree for the user and provides all of the necessary functionality for solving linear and mixed-integer programs. Algorithm 3 takes a single input parameter: `MIPmodel`. Initially, `MIPmodel` represents the MIP formulation (19a)–(19e). The only advanced technique that we employ is in telling the solver how to identify if a violated EP constraint (19f) exists. This step is implemented through a "callback" function, which virtually all of the leading commercial and open-source MIP solvers provide.

ALGORITHM 3 `BranchAndCutforMTTA` *(*`MIPmodel`*)*

1: Write a callback function to do the following at each node in the search tree:
2: **while** $\hat{\theta} < f(\hat{\mathbf{y}})$ **do**
3: Call `GreedyAlgorithm`$(f, \hat{\mathbf{y}})$ and append the returned EP inequality $\pi^T\mathbf{y} \leq \theta - f(\emptyset)$ to `MIPmodel`
4: Re-solve the LP relaxation of `MIPmodel` to produce an updated partial solution $(\hat{\theta}, \hat{\mathbf{y}})$
5: **end while**
6: Invoke the MIP solver's `solve()` method with the callback function
7: **return** The single best track-to-track association solution $\hat{\mathbf{y}}^*$ to `MIPmodel`

When solving a MIP, all leading MIP solvers relax the binary constraints $\mathbf{y} \in \{0,1\}^n$ with linear constraints $\mathbf{y} \in [0,1]^n$ so that a linear program (LP) is solved at every node in the search tree. Before branching on a decision variable that is required to be integral, but is currently fractional at the optimal solution of the LP relaxation, the solver will attempt to generate a number of default cuts that apply to all MIPs as well as those cuts that the user requested the solver to attempt to generate through a callback function. Thus, if in Step 2 the solver finds that $\hat{\theta} < f(\hat{\mathbf{y}})$, the most violated EP cut (19f) is appended to `MIPmodel` and the LP relaxation of

[7] Up to this point, we have described f as a set function, i.e., as a function $f(S)$ whose argument is a subset S of N or, equivalently, as a function $f(\mathbf{y})$ whose argument is a binary vector $\mathbf{y} \in \{0,1\}^n$. Henceforth, when we write $f(\mathbf{y})$, we will treat f as a continuous function of $\mathbf{y} \in [0,1]^n$.

MIPmodel is re-solved in Step 4 for an updated partial solution $(\hat{\theta}, \hat{\mathbf{y}})$. In the event that the node being processed is a leaf node, which implies that $\hat{\mathbf{y}}$ is binary, the solver will still call the callback function (Step 2) to check for violated EP inequalities. This loop is repeated at every node until no more violated EP inequalities are found. When Algorithm 3 has terminated, MIPmodel will likely have a very small subset of EP constraints as well as a number of default MIP cuts that were added by the solver. This is an important observation because we will re-use this updated MIPmodel when finding the K best solutions.

THEOREM 3 (Algorithm 3 is exact) *The proposed branch-and-cut procedure finds the provably best (globally optimal) solution to the MTTA problem* (11) *when the input parameter* MIPmodel *represents the MIP formulation* (19a)–(19e).

PROOF Let $(\theta^*, \mathbf{w}^*, \mathbf{y}^*, \mathbf{z}^*)$ be an optimal solution to (19) with corresponding objective function value γ^*. Let $(\theta^{**}, \mathbf{w}^{**}, \mathbf{y}^{**}, \mathbf{z}^{**})$ be the solution returned by Algorithm 3 with corresponding objective function value γ^{**}. Note that $\gamma^{**} \leq \gamma^*$since, upon termination, MIPmodel does not necessarily (and most likely does not) contain all EP constraints (19f) and is, therefore, a relaxation of (19). Meanwhile, $(\theta^{**}, \mathbf{w}^{**}, \mathbf{y}^{**}, \mathbf{z}^{**})$ is feasible to (19) since in Step 3 no more violated EP constraints could be found, which by Corollary 1 occurs if and only if $(\theta^{**}, \mathbf{y}^{**}) \in Q_f$. Thus, $(\theta^{**}, \mathbf{w}^{**}, \mathbf{y}^{**}, \mathbf{z}^{**})$ must be an optimal solution to (19) and the assignment $\mathbf{y}^{**}$is an optimal solution to the MTTA problem (11).

Thus far, we have described how to find the single provably best solution to the MTTA problem. In order to find the K provably best solutions, we essentially make K calls to Algorithm 3. A high-level description of the algorithm for finding the K provably best solutions to the MTTA problem is presented in Algorithm 4. In Step 1, we formulate MIPmodel as the MIP formulation (19a)–(19e). It is important to note that MIPmodel will be modified throughout the algorithm as other constraints are appended to it. Step 2 is not necessary, but is almost always a good idea since MIP solvers generally perform better when good incumbent solutions are already available. The main **while** loop in Steps 4–10 requires the most computation time. Initially, we find the kth best solution for $k = 1$. In general, after finding the kth best optimal solution $\hat{\mathbf{y}}^k$, we add a so-called *enumeration cut*

$$\sum_{j=1:\hat{y}_j^k=1}^{n} (1 - y_j) + \sum_{j=1:\hat{y}_j^k=0}^{n} y_j \geq 1 \tag{20}$$

to MIPmodel to render the current optimal solution infeasible and thus allow us to find the solution corresponding to the next best association. Re-solving MIPmodel in Step 5 starting from the formerly optimal, but now infeasible, solution will yield the next best provably optimal solution.

ALGORITHM 4 MIPforMTTA(K)

1: Formulate the MIP (19a)–(19e). Call this formulation MIPmodel.
2: Obtain high-quality incumbent solutions by solving the GNPM problem for the K best bias-assignment solutions $\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^K$ using a good heuristic
3: Set numSolnsFound=0
4: **while** numSolnsFound $< K$ **do**
5: Call BranchAndCutforMTTA (MIPmodel) for an optimal assignment $\hat{\mathbf{y}}^*$ to MIPmodel
6: Update the list of the K best solutions, if possible
7: numSolnsFound++
8: **if** numSolnsFound $< K$ **then**
9: Append an enumeration cut to MIPmodel: $\sum_{j=1:\hat{y}_j^*=1}^{n}(1 - y_j) + \sum_{j=1:\hat{y}_j^*=0}^{n} y_j \geq 1$
10: **end if**
11: **end while**
12: **return** The K best track-to-track association solutions $\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^K$

THEOREM 4 (Algorithm 4 is exact) *Algorithm 4 finds the K provably best (globally optimal) solutions to the MTTA problem* (11).

PROOF By induction. Let k = numSolnsFound. When $k = 0$, Algorithm 4 calls Algorithm 3 without any enumeration cuts appended to MIPmodel. By Theorem 4, the solution returned by Algorithm 3 without any enumeration cuts is a provably optimal solution to (19). For the inductive step, assume that at the start of Step 4 when $0 < k < K$, the k provably best associations $\mathbf{y}^1, \ldots, \mathbf{y}^k$ have been found and that MIPmodel includes k enumeration cuts (20) that render the k best associations infeasible. Since all solutions to the MIP (19a)–(19e) are still feasible to MIPmodel, except for those whose $\mathbf{y}$-component corresponds to one of the k best associations, the next call to BranchAndCutforMTTA(MIPmodel) will solve a MIP over a smaller feasible region and, by using arguments virtually identical to those given in the proof of Theorem 4, will return a solution whose $\mathbf{y}$-component corresponds to the $(k + 1)$th globally optimal association.

7. PERFORMANCE STUDIES

The purpose of this section is to compare the performance of the two exact algorithms described in the previous section. It will be shown that the branch-and-cut algorithm (Approach 2) is vastly superior to the branch-and-bound method (Approach 1). We will also see empirical evidence that the K best MTTA solutions are often a subset of the $3K$ best GNPM solutions. It is for this reason, as well as for the arguments given in Section 4 with the illustrative example, that we do not specifically evaluate the suitability of the MTTA likelihood function for performing track-to-track association in comparison to the GNPM likelihood function. The

TABLE IX
Density Guidelines

Overlap %	MMD	PCA
100%	[0,2)	[0,0.75)
	[2,4)	[0.75,0.95)
	[4,∞)	[0.95,1]
70%	[0,4)	[0,0.75)
	[4,8)	[0.75,0.95)
	[8,∞)	[0.95,1]

TABLE X
3D Cube Side Lengths

	Density		
n_B	High	Medium	Low
5	5.26	10.51	15.77
6	6.10	12.20	18.29
7	6.89	13.79	20.68
8	7.65	15.29	22.94
9	8.36	16.72	25.08
10	9.06	18.11	27.17
11	9.72	19.44	29.16
12	10.37	20.73	31.10
13	10.99	21.98	32.97
14	11.61	23.22	34.83
15	12.20	24.39	36.59

superiority of one over the other is an open question, although we contend that the MTTA likelihood function is more suitable in the case of a one-time handover.

7.1. Instance Generator

For ease of explanation and reproducibility for other researchers, we present computational results for 3-dimensional "box" instances of the MTTA problem. Although numerous experiments involving more realistic tracking scenarios in 6 dimensions have been completed, we believe that the same insights about the algorithms can be gleaned from these contrived instances.

Track states are maintained in a 3-dimensional Cartesian reference frame. All covariance matrices are diagonal matrices: $\mathbf{R} = \mathbf{P}_i = \mathbf{Q}_j = \mathbf{I}$ so that $\mathbf{S}_{ij} = 2\mathbf{I}$, $\forall i \in N_A$, $\forall j \in N_B$. The symmetry (i.e., the homogeneity of the covariances matrices) of these instances leads to difficulties for the algorithms as they cannot help distinguish between choosing one track pairing over another. So while these instances may appear simple, they are actually more difficult than most instances encountered in practice in which disparate covariances are typical.

A particular "box" instance is created as follows: After choosing n_A and n_B (with $n_A \leq n_B$), n_B objects are created by randomly generating position components uniformly in a cube with a given side length and assigned to sensor B. Next, n_A of the n_B objects are identified and assigned to sensor A. The true bias is drawn randomly from a Gaussian$(\mathbf{0},\mathbf{R})$ distribution and is added to each track in N_A. Finally, each track in N_A and N_B receives a random measurement error, drawn randomly from a Gaussian$(\mathbf{0},\mathbf{I})$ distribution.

The side length of the cube from which the position estimates of the tracks are randomly drawn influences the track scene density, i.e., how closely spaced the objects are to one another. In general, the more closely spaced are the tracks, the more difficult it is to correctly associate tracks (with respect to truth) and the more computational effort is required to find provably optimal solutions. While many metrics could be used to gauge scene density, one that has been used in a number of studies is the *minimum Mahalanobis positional distance* (MMD) computed over all tracks in each track set. It is a unitless metric sometimes referred to as the minimum nearest neighbor distance. Based on several computational studies, e.g., [25], [23], [7], Table IX was constructed to provide rough guidelines regarding the difficulty to correctly associate tracks in terms of the probability of correct association (PCA), defined to be the fraction of the n_A sensor A tracks that the association algorithm correctly associates based on truth. For example, when $n_A = n_B$ (100% overlap between the two track sets) and the MMD is less than 2, Table IX indicates that PCA is, on average, less than 0.75. Similar guidelines are given for an overlap of 70%, meaning $n_A = 0.7n_B$.

Using the above density guidelines, we evaluated the performance of our algorithms at three different scene densities: low, medium, and high. A high density scene corresponds to a MMD in the interval [1,2), a medium density in [2.5,3.5], and a low density in [4,5]. For low, medium, and high densities, we expected and confirmed that the average PCA was at least 0.95, between 0.75 and 0.95, and less than 0.75, respectively. To generate instances with these densities, we generated n_B points from a cube with the side length given in Table X. These parameters were computed through simulation and yield, on average, a MMD in the desired interval. If the MMD did not fall within the desired interval, this instance was discarded and new instances were generated until the MMD fell within the desired interval.

7.2. Computational Results

All computations were carried out on a Linux machine with kernel 2.6.18 running on a 64-bit x86 processor and 32GB of RAM. For every choice of parameter settings tested, 100 Monte Carlo experiments were performed. Both algorithms were implemented in Java and the MIP solver of CPLEX 11.2 [13] was used for the branch-and-cut algorithm. A solution was declared optimal if the relative optimality gap $(z^* - z^{LB})/z^*$ was at most 0.0001, where z^* is the value of the current best solution, and z^{LB} is a lower bound on the value of the MTTA problem. At the start of each exact algorithm, we employed the "All-Pairs" heuristic proposed in [5] to obtain K near-optimal solutions to the GNPM problem. While there are many heuristics available, we chose the

TABLE XI
Branch-and-Bound CPU Times (seconds)

# Tracks		Density		
n_A	n_B	Low	Medium	High
5	5	1.6981	1.8756	2.3923
6	6	16.5326	19.6353	25.1325
7	7	123.4889	131.6625	213.7652
8	8	287.4124	294.4444	—
9	9	—	—	—
10	10	—	—	—
5	10	60.7861	69.8828	87.3136
6	11	142.4322	155.6128	—
7	12	259.7773	274.3616	—
8	13	—	—	—
9	14	—	—	—
10	15	—	—	—

TABLE XII
Branch-and-Cut CPU Times (seconds)

# Tracks		Density		
n_A	n_B	Low	Medium	High
5	5	0.3599	0.4417	1.4738
6	6	0.4976	0.6367	1.5398
7	7	0.6778	0.7988	4.0195
8	8	0.9395	1.1405	7.1438
9	9	1.1608	1.4496	10.3954
10	10	1.5551	1.8486	23.5589
11	11	1.9391	2.2815	29.5981
12	12	2.6192	2.7251	31.3729
13	13	3.2803	3.3583	56.5361
14	14	4.1458	4.2619	74.1426
15	15	4.4676	4.8745	78.0808
5	10	0.4654	0.5369	1.5663
6	11	0.6787	0.7290	2.9092
7	12	0.8933	0.9331	4.2180
8	13	1.1667	1.2309	6.4942
9	14	1.5173	1.5502	10.0925
10	15	1.9460	1.9844	12.3069

"All-Pairs" heuristic for its simplicity and deterministic behavior. As will be explained in Fig. 5, we have empirical evidence that the GNPM likelihood function closely approximates the MTTA likelihood function (at least in the region of optimal solutions), and so the K near-optimal GNPM solutions returned by the heuristic are near-optimal for the MTTA problem. This is good news since it means that good heuristics already exist and new heuristics do not need to be developed. Since the heuristic is typically able to find a subset of the K best solutions, most of the effort carried out by the exact algorithms is in proving that these solutions are, indeed, optimal.

Tables XI and XII show the average solution times in seconds needed for the branch-and-bound and the branch-and-cut algorithms, respectively, to identify the $K = 10$ provably best solutions for various track sizes and scene densities. A dash (—) in the table means that, on average, an algorithm did not terminate within a five minute time limit. Even after significant experimentation with different parameter settings, the branch-and-bound method is vastly inferior to the branch-and-cut algorithm. Indeed, the branch-and-bound method cannot solve instances with more than 8 tracks in each set in under five minutes. Meanwhile, for low and medium scene densities, the branch-and-cut algorithm performs remarkably well with average solution times under five seconds. The standard deviation in solution time was also under a second for each parameter setting.

Although the solution times for the high density track scenes may appear discouraging, empirically we have observed that any association algorithm optimizing with respect to the GNPM or MTTA likelihood function will have a low PCA at such a high density. In fact, these track scenes are so congested that one is likely to question the fidelity of the tracks that were formed, i.e., the observation-to-track assignments that were made to produce the tracks are likely to be flawed. In light of this caveat, the solution times presented for the high density track scenes should be considered a "stress test" for the algorithm under extreme conditions. And if these conditions are encountered in practice, then one might verify that the track states and covariances are valid.

We attribute this marked discrepancy in performance between the two algorithms to two facts. First, in the branch-and-bound method, the QAP relaxations at shallow depths in the search tree are poor and do not permit early pruning, which leads to many nodes being expanded early in the search. Second, once the bounds do become useful for pruning, one still has to solve a small QAP, which may require roughly a second of computation time. Coupling these two facts, we see that many small QAPs, each requiring a small, but non-negligible amount of time, quickly adds up. On the other hand, the branch-and-cut method fully exploits the power and efficiencies that are now standard in MIP solvers, which leads to impressive computation times.

Another interesting question is: How well does the branch-and-cut algorithm scale as the user requests more and more provably optimal solutions, i.e., as the parameter K increases? Fig. 4 illustrates that, for various track sizes with a medium scene density, the algorithm scales almost linearly in K.

Fig. 5 shows the approximate number of provably optimal GNPM solutions that must be generated in order to ensure, with a high probability, that the K provably best solutions are obtained. From a theoretical perspective, this figure shows the close relationship between the GNPM and MTTA likelihood functions. From a practical perspective, this figure is important because it indicates that a user who does not wish to implement the branch-and-cut method, or any other exact algorithm for that matter, but who already has access to a good heuristic for the GNPM problem, can use the existing heuristic to generate optimal or near-optimal solutions with high

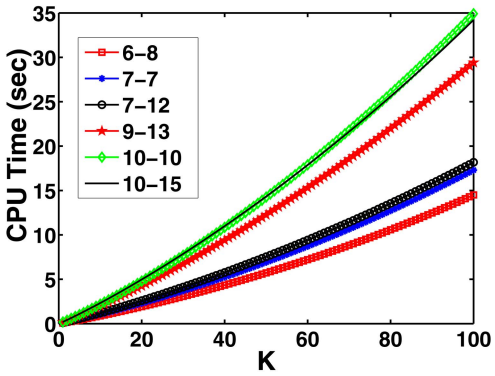


Fig. 4. Solution time (seconds) as a function of K.

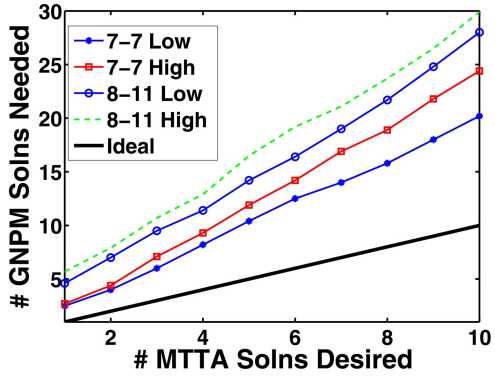


Fig. 5. Number of GNPM solutions needed.

confidence. Indeed, after choosing the number K of optimal MTTA solutions desired on the x-axis, the user can get a rough sense of the number of GNPM solutions that should be generated in order to be confident that the K best MTTA solutions have been found. Specifically, for a given choice of K, Fig. 5 shows the mean number (plus three standard deviations) of provably optimal GNPM solutions needed to obtain the K provably best MTTA solutions. Ideally, the K best MTTA hypotheses would coincide with the K best GNPM hypotheses in which case the fast multistart local search heuristics that are already available for the GNPM problem could be used for the MTTA problem without any loss of optimality. In this ideal setting, we would obtain the thick black line emanating from the origin. Unfortunately, the best hypotheses for the different likelihood function generally do not move in lock step.

Although we only present results up to the 10 best solutions, we have found that this pattern continues. Thus, we suggest that if a user wishes to obtain the K best MTTA solutions with high confidence without implementing one of our exact algorithms, he should find the $3K$ best GNPM solutions to achieve this goal.

At various times, we have used the superlative "remarkable" to describe the branch-and-cut algorithm. We close this section to explain why we believe the results for the branch-and-cut algorithm are so impressive. First, operational requirements for most association algorithms typically allow at most a few seconds of computation time. Thus, having the ability to return provably optimal solutions in such a small time window is a desirable feature. Second, since many heuristics for the GNPM problem require one- or two-tenths of a second to return a number of near optimal solutions, it appears that the increase in computation time to identify provably optimal solutions is roughly an order of magnitude. Given that the increase in solution time to find a single provably optimal solution over a heuristic solution is often several orders of magnitude for many difficult 0-1 optimization problems, having an order of magnitude increase is good news.

8. CONCLUSIONS

The primary goal of this paper was to introduce a marginal track-to-track association likelihood function for track ambiguity management, which takes into account all of the major issues considered by other popular association likelihood functions, but is more suitable for system-level track ambiguity management, especially for a one-time handover. We described how pairwise track-to-track likelihoods could be constructed to quantify the confidence in pairing two tracks together. Our final contribution was the introduction of two exact algorithms that can solve a track-to-track association problem using the likelihood function that we introduced. The second approach, which exploits well-known results from submodular function minimization, performs quite well.

As there is on-going effort to develop efficient and robust algorithms for performing track-to-track association between more than two sensors, we believe that understanding the possible types of information that can be extracted when only two sensors are participating and the shortcomings of the related algorithms to obtain this data are crucial when considering more complex multisensor problems. In addition, since data association is a low level step in the data fusion process, improving metrics and algorithms for track correlation can have a significant impact on system performance.

ACKNOWLEDGMENTS

We wish to thank Arjang Noushin, John-David 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 (8). We thank three anonymous referees for their insightful comments which helped improve the paper as a whole.

REFERENCES

[1] A. Atamtürk and V. Narayanan
Polymatroids and mean-risk minimization in discrete optimization.
Operations Research Letters, **36** (2008), 618–622.

[2] Y. Bar-Shalom and H. Chen
Track-to-track association using attributes.
Journal of Advances in Information Fusion, **2**, 1 (2007), 49–59.

[3] S. S. Blackman and R. Popoli
Design and Analysis of Modern Tracking Systems.
Norwood, MA: Artech House, 1999.

[4] E. Çela
The Quadratic Assignment Problem: Theory and Algorithms.
Kluwer Academic Publishers, 1998.

[5] S. Danford, B. D. Kragel, and A. B. Poore
Joint bias estimation and data association: Algorithms.
In O. E. Drummond and R. D. Teichgraeber (Eds.), *Proceedings of SPIE*, vol. 6699, Signal and Data Processing of Small Targets, 2007.

[6] J. Edmonds
Submodular functions, matroids and certain polyhedra.
In R. Guy (Ed.), *Combinatorial Structures and Their Applications*, vol. 11, Gordon and Breach, New York, NY, 1971, 69–87.

[7] T. Fercho and D. J. Papageorgiou
Feature-aided global nearest pattern matching with non-gaussian feature measurement errors.
In *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT: IEEE, Mar. 2009.

[8] J. Ferry
Exact bias removal for the track-to-track association problem.
In *Proceedings of the 12th International Conference on Information Fusion*, Seattle, WA, July 2009.

[9] J. Ferry
Exact association probability for data with bias and features.
Journal of Advances in Information Fusion, **5**, 1 (2010), 41–67.

[10] D. L. Hall and J. Llinas
An introduction to multisensor data fusion.
Proceedings of the IEEE, **85**, 1 (1997), 1–18.

[11] R. E. Helmick and T. R. Rice
Removal of alignment errors in an integrated system of two 3-d sensors.
IEEE Transactions on Aerospace and Electronic Systems, **29**, 4 (1993), 1333–1343.

[12] S. M. Herman and A. B. Poore
Nonlinear least-squares estimation for sensor and navigation biases.
In O. E. Drummond (Ed.), *Proceedings of SPIE*, vol. 6236, Signal and Data Processing of Small Targets, Aug. 2006.

[13] ILOG CPLEX
Reference Manual, Version 11.2, 2008.

[14] R. J. Kenefic
Local and remote track file registration using minimum description length.
IEEE Transactions on Aerospace and Electronic Systems, **29**, 3 (1993), 651–655.

[15] B. D. Kragel, S. Danford, S. M. Herman, and A. B. Poore
Bias estimation using targets of opportunity.
In O. E. Drummond (Ed.), *Proceedings of SPIE*, vol. 6699, Signal and Data Processing of Small Targets, Aug. 2007.

[16] B. D. Kragel, S. Danford, and A. B. Poore
Concurrent map data association and absolute bias estimation with an arbitrary number of sensors.
In O. E. Drummond (Ed.), *Proceedings of SPIE*, vol. 6969, Signal and Data Processing of Small Targets, Aug. 2008.

[17] M. Levedahl
An explicit pattern matching assignment algorithm.
In O. E. Drummond (Ed.), *Proceedings of SPIE*, vol. 4728, Signal and Data Processing of Small Targets, Aug. 2002, 461–469.

[18] L. Lovász and A. Schrijver
Cones of matrices and set-functions and 0-1 optimization.
SIAM Journal on Optimization, **1** (1991), 166–190.

[19] J. R. Moore and W. D. Blair
Practical aspects of multisensor tracking.
In Y. Bar-Shalom and W. D. Blair (Eds.), *Multitarget-Multisensor Tracking: Applications and Advances*, Norwood, MA: Artech House, 2000, vol. III, ch. 2.

[20] S. Mori and C-Y. Chong
Effects of unpaired objects and sensor biases on track-to-track association: Problems and solutions.
In *Proceedings of MSS National Symposium on Sensor and Data Fusion*, vol. 1, 2000, 137–151.

[21] S. Mori and C-Y. Chong
Comparison of bias removal algorithms in track-to-track association.
In O. E. Drummond and R. D. Teichgraeber (Eds.), *Proceedings of SPIE*, vol. 6699, Signal and Data Processing of Small Targets, 2007.

[22] G. L. Nemhauser and L. A. Wolsey
Integer and Combinatorial Optimization.
New York: Wiley, 1988.

[23] D. J. Papageorgiou
Track-to-track association using pure association likelihoods.
Raytheon Technical Report, Aug. 2008.

[24] D. J. Papageorgiou and M. Holender
Track-to-track association and ambiguity management in the presence of sensor bias.
In *Proceedings of the 12th International Conference on Information Fusion*, Seattle, WA, July 2009.

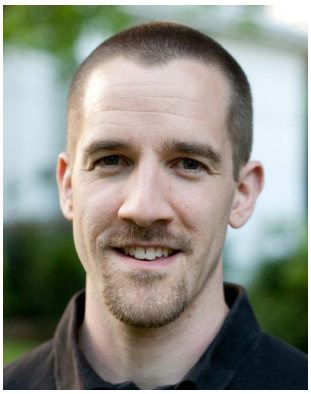
[25] D. J. Papageorgiou and J-D. Sergi
Simultaneous track-to-track association and bias removal using multistart local search.
In *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT: IEEE, Mar. 2008.

[26] H. Sherali and W. Adams
A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems.
SIAM Journal on Discrete Math, **3**, 3 (1990), 411–430.

[27] L. D. Stone, M. L. Williams, and T. M. Tran
Track-to-track association and bias removal.
In O. E. Drummond (Ed.), *Proceedings of SPIE*, vol. 4728, Signal and Data Processing of Small Targets, Aug. 2002, 315–329.

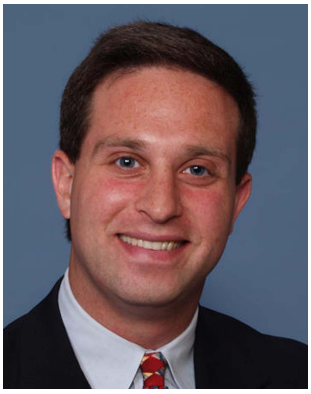
[28] B. Zhou and N. K. Bose
An efficient algorithm for data association in multitarget tracking.
IEEE Transactions on Aerospace and Electronic Systems, **31**, 1 (1995), 458–468.

[29] Y. Zhu
Recent advances and challenges in quadratic assignment and related problems.
Ph.D. dissertation, University of Pennsylvania, 2007.



Dimitri Papageorgiou holds a B.S. in mathematics from the University of North Carolina at Chapel Hill and an M.S. in operations research and industrial engineering from the University of Texas at Austin. He is currently finishing his Ph.D. in operations research under the aegis of George Nemhauser in the H. Milton Stewart School of Industrial and Systems Engineering at the Georgia Institute of Technology.

He is a recipient of a U.S. National Science Foundation Graduate Research Fellowship and was the only person in the country pursuing a doctorate in the field of operations research to receive this honor in his year. As a former member of Raytheon's Center for Discrimination, he has been involved in projects related to sensor resource management, multi-target tracking, and data association, and has co-authored several papers in these areas.



Michael Holender holds a B.S. in mathematics and statistics from Miami University in Oxford, OH as well as an M.S. in industrial and systems engineering from SUNY at Buffalo. He earned his Ph.D. in industrial and systems engineering (operations research) from the State University of New York (SUNY) at Buffalo while studying information fusion.

He is currently a senior systems engineer with the Integrated Defense Systems Division of The Raytheon Company where he has been involved in multi-target tracking and discrimination algorithms. He helped design a new set of algorithms for simultaneous association and classification of objects rooted in the theory of conceptual spaces.