

# Joint MAP Bias Estimation and Data Association: Algorithms

Scott Danford, Bret Kragel, and Aubrey Poore  
Numerica Corporation  
P.O. Box 271246  
Fort Collins, CO 80527-1246

## ABSTRACT

The problem of joint maximum a posteriori (MAP) bias estimation and data association belongs to a class of nonconvex mixed integer nonlinear programming problems. These problems are difficult to solve due to both the combinatorial nature of the problem and the nonconvexity of the objective function or constraints. A specific problem that has received some attention in the tracking literature is that of the target object map problem in which one tries match a set of tracks as observed by two different sensors in the presence of biases, which are modeled here as a translation between the track states. The general framework also applies to problems in which the costs are general nonlinear functions of the biases.

The goal of this paper is to present a class of algorithms based on the branch and bound framework and the “all-pairs” and k-best heuristics that provide a good initial upper bound for a branch and bound algorithm. These heuristics can be used as part of a real-time algorithm or as part of an “anytime algorithm” within the branch and bound framework. In addition, we consider both the A\*-search and depth-first search procedures as well as several efficiency improvements such as gating. While this paper focuses on the algorithms, a second paper will focus on simulations.

**Keywords:** MAP Bias Estimation, Data Association, Heuristics, A\*-Search, Branch and Bound, K-Best Solutions, Nonconvex MINLP

## 1. INTRODUCTION

Bias estimation and residual bias mitigation are prerequisites for successful multiple sensor data fusion. Bias estimates can be based on filtering or batch least squares with or without process noise. For training data in the estimation problem, one frequently considers truth data, which may or may not be certain, or targets of opportunity for which the data association is known. In addition, some biases are observable and others are not, depending to some extent on the training data. Bias estimates can also be classified as either absolute or relative as discussed in the paper by Herman.<sup>1</sup> Once estimates are determined and the sensor reports and navigation errors are corrected, one must then address the residual bias problem, i.e., the remaining uncertainty in the stochastic bias estimates after measurements and transformations are corrected.<sup>2</sup>

At times, truth objects or targets of opportunity may not be available or may not be sufficient to yield maximum observability of the biases. In this case, one might consider turning to the targets themselves for which data association is unknown. The development of a method for using objects with unknown data association is a difficult one; however, much progress has been achieved in this direction for what is called the target object map problem as presented in the book by Blackman and Popoli [3, Section 10.5.5] and as further addressed by Levedahl’s global nearest pattern algorithm<sup>4,5</sup> and references therein. This joint MAP bias estimation and association problem as formulated in Section 2 is perhaps the simplest of the problems that combine data association and maximum a posteriori (MAP) bias estimation; however, it is already difficult in that it is a nonconvex, mixed integer nonlinear programming (MINLP) problem. On the other hand, MINLP problems have been addressed extensively in the operations research literature<sup>6-9</sup> and we make use of some of the corresponding computational experience. Techniques for solving convex MINLP problems include outer approximation, generalized Bender’s decomposition, extended cutting plane, branch and bound, and branch and cut methods. For the nonconvex case,

one attempts to extend these methods by using convex envelopes and convex under-estimators of the objective function and constraints. In addition, sampling methods, such as clustering methods, evolutionary algorithms, simulated annealing, and tabu search, are also appropriate.

In particular, Levedahl<sup>5</sup> has developed an assignment algorithm that specifically allows optimal assignments in the presence of a persistent bias between two sets of data. The algorithm is based upon a Dijkstra search after organizing the assignment problem hypothesis space into a tree and, as the search is based upon dynamic programming, the solution found is guaranteed to be optimal. Levedahl’s algorithm inherently supports finding the k-best rather than just the single best solution, and has proven to be significantly faster than approaches such as the “all-pairs” algorithm (Section 3.2) using an auction or JVC algorithm applied to a log-likelihood or chi-squared cost matrix, while also achieving much higher assignment accuracy in the presence of significant bias. Levedahl’s algorithm can also be viewed as an A\*-search algorithm.

In this work, we pursue a different class of algorithms based on the development of a good initial heuristic followed by the application of one of a class of branch and bound algorithms for constructing a guaranteed optimum. A good heuristic can provide an excellent upper bound for the branch and bound procedure. Thus, the initial focus will be to review a commonly used local search algorithm for improving a nonoptimal solution and to explain the “all-pairs” heuristic, which can provide very good solutions, but does not have good runtime complexity. To improve the runtime performance and maintain the quality of the “all-pairs” heuristic, we develop a new heuristic called the “k-best” heuristic as presented in Section 3.

With respect to the class of branch and bound algorithms developed herein, one must recognize that in the worst case, the time required to solve to a guaranteed optimal solution can grow exponentially in the size of the problem. In spite of this, there is support for this approach found in the literature when solving nonconvex MINLP problems to optimality. Borchers and Mitchell<sup>10</sup> compared an experimental branch and bound code with a commercially available outer approximation code on a number of test problems and found that the two were roughly comparable in speed and robustness. For convex quadratic problems, Fletcher and Leyffer<sup>11</sup> compared the performance of their branch and bound code with their implementations of outer approximation, generalized Bender’s decomposition, and an algorithm that combines branch and bound and outer approximation approaches. Their conclusion was that their branch and bound solver was consistently faster than their other algorithms by an order of magnitude. Finally, Zhang<sup>12</sup> demonstrated that branch and bound can also be used as a good “anytime” algorithm in a case study of depth-first branch and bound versus local search.

One of the key obstacles in the development of branch and bound algorithms for nonconvex problems is the requirement of a global lower bound for each node. That is the reason for the development of the aforementioned convex underestimators. We accomplish this convex underestimator and thus achieve the required global lower bound by zeroing out the Mahalanobis distance part of the costs as explained in Section 4.1.

There are many possible choices in the branch and bound algorithm including methods for the construction of the upper and lower bound for a node, the partitioning method and the search procedure. Since the efficiency of the branch and bound procedure depends extensively on the search procedure, we explain two of these, namely, A\*-search and a depth-first search. We discuss several computational efficiencies such as gating, recursive estimation, and problem decomposition. Also, the computation of the k-best solutions is easily accommodated within the branch and bound framework as discussed in Section 4.7.

The algorithm that we prefer is the combination of the k-best heuristic and a partial or full branch and bound. This heuristic has excellent runtime performance and produces a near-optimal solution, which in turn provides an excellent upper bound for a branch and bound procedure. One can either run the branch and bound procedure to completion for guaranteed optimality or can stop it at anytime with what is generally a solution better than the heuristic.

The A\*-search algorithm developed within the branch and bound framework can stand alone as a separate algorithm. We find that the ability to fathom nodes can be a benefit to A\*-search, but at an additional computation. In addition, the first k-leaf nodes explored or expanded in this algorithm are k-best solutions.

Finally, while the posed problem is based on the two dimensional assignment formulation, the general framework can be extended to the multidimensional assignment problem used in multiple hypothesis tracking. A

statement of the costs is given in the appendix and a full derivation, in the work of Kragel, Danford, Herman, and Poore.<sup>13</sup>

While this paper focuses on algorithms, a second paper<sup>14</sup> provides simulation results to support some of the theoretical algorithm conclusions. The paper is organized as follows. The formulation of the joint MAP bias estimation and data association problem investigated here is presented in the Section 2. The appendix contains a general N-source formulation corresponding to this two-source problem. Section 3 describes the local search algorithm and presents an overview of heuristics including the “all-pairs” algorithm and those based on k-best intersection and union algorithms. The branch and bound algorithm is presented in Section 4.

## 2. THE MAP BIAS ESTIMATION-ASSOCIATION PROBLEM

Consider an unknown number  $N$  of objects moving independently in space, some of which are being tracked by two sensors, with sensor 1 having tracks on  $m \leq N$  objects, while sensor 2 is tracking  $n \leq N$  objects. The sensor tracks for each object consist of an estimate of the true, but unknown, kinematic state of the object, along with a covariance matrix describing the stochastic uncertainty in the estimate. In general, the number of objects tracked by each sensor will be different, i.e.,  $m \neq n$ , and it will often be the case that each sensor has tracks on some objects not being tracked by the other sensor. We assume that the track states are transformed to a common reference frame and are time aligned. We consider three cases: (a) a prior bias estimate on each sensor, (b) no bias prior, and (c) a combined bias prior, where case (a) can be subdivided into several additional cases.

We have derived a formulation of the multidimensional version in<sup>13</sup> with a brief statement of the results in the Appendix A.

### 2.1 Case of a Bias Prior Per Sensor

The sensor tracks are represented by a probability distribution on the corresponding target state estimates:

$$\begin{aligned}\hat{x}_i &= x_i - d_x + \mu_i, & i &= 1, \dots, m \\ \hat{y}_j &= y_j - d_y + \nu_j, & j &= 1, \dots, n,\end{aligned}$$

where for sensor 1,  $\hat{x}_i$  denotes the estimated kinematic state of track  $i$ ,  $x_i$  is the corresponding true, but unknown, kinematic state,  $d_x$  is an unknown bias offset vector and  $\mu_i$  denotes the random estimation error. Likewise, for sensor 2,  $\hat{y}_j$  denotes the estimated kinematic state of track  $j$ ,  $y_j$  is the corresponding true, but unknown, kinematic state,  $d_y$  is an unknown bias offset vector and  $\nu_j$  denotes the random estimation error. The random estimation errors  $(\mu_i^T \quad \nu_j^T)^T$  may be correlated so we assume they are zero-mean Gaussian with covariance  $E[\mu_i \nu_j^T] = S_{ij}$ .

The problem to be addressed in this work is that of determining which of the tracks from sensor 1 and sensor 2 correspond to the same object, while concurrently computing estimates  $d_x$  and  $d_y$  of the relative bias  $d = d_x - d_y$  by utilizing their covariance

$$R = \begin{pmatrix} R_{xx} & R_{xy} \\ R_{yx} & R_{yy} \end{pmatrix} = E \left[ \begin{pmatrix} d_x \\ d_y \end{pmatrix} \begin{pmatrix} d_x \\ d_y \end{pmatrix}^T \right],$$

which is assumed to be known. The joint MAP bias estimation and data association problem may be put in the form of a modified general network flow problem as

$$\begin{aligned}
& \text{Minimize}_{(x,d_x,d_y)} c_r(d_x, d_y) + \sum_{(i,j) \in A} c_{ij}(d_x, d_y) x_{ij} \\
& \text{Subject To: } \sum_{j \in A(0)} x_{0j} = n, \\
& \sum_{j \in A(i)} x_{ij} = 1 \quad (i = 1, \dots, m), \\
& \sum_{i \in B(0)} x_{i0} = m, \\
& \sum_{i \in B(j)} x_{ij} = 1 \quad (j = 1, \dots, n), \\
& x_{ij} \in \begin{cases} \{0, 1\} & \text{for } i \neq 0 \text{ or } j \neq 0, \\ \{0, 1, \dots, \text{Min}\{m, n\}\} & \text{for } i = 0 \text{ and } j = 0, \end{cases}
\end{aligned} \tag{1}$$

where  $A$  is the set of feasible arcs,  $A(i) = \{j \mid (i, j) \in A\}$  and  $B(j) = \{i \mid (i, j) \in A\}$ , such that  $A(0) = J$ ,  $B(0) = I$ , and

$$c_r(d_x, d_y) = \frac{1}{2} \begin{pmatrix} d_x \\ d_y \end{pmatrix}^T \begin{pmatrix} R_{xx} & R_{xy} \\ R_{yx} & R_{yy} \end{pmatrix} \begin{pmatrix} d_x \\ d_y \end{pmatrix} \tag{2}$$

$$\begin{aligned}
c_{ij}(d_x, d_y) &= \frac{1}{2} (\hat{x}_i + d_x - (\hat{y}_j + d_y))^T Z_{ij}^{-1} (\hat{x}_i + d_x - (\hat{y}_j + d_y)) \\
&\quad + \frac{1}{2} \ln [\beta_T^2 |2\pi Z_{ij}|] + \gamma_{ij} \text{ for } i \neq 0 \text{ and } j \neq 0,
\end{aligned} \tag{3}$$

$$Z_{ij} = P_{ii} - S_{ij} - S_{ji} + Q_{jj} \tag{4}$$

where  $\beta_T$  denotes the target density,  $\gamma_{ij} = \ln((1 - P_D^1)(1 - P_D^2))$ , and  $P_D^k$ ,  $k \in \{1, 2\}$  is the probability of sensor  $k$  observing a particular target.

Similar to the book by Blackman and Popoli<sup>3</sup> and publications by Levedahl,<sup>4,5</sup> the terms involving the registration covariance matrix  $R$  serve to include a prior estimate of the bias in the objective function. Note also that the singleton costs  $c_{i0} = c_{0j} = 0$  are independent of  $d$ .

## 2.2 The Estimation Problem

The solution of the estimation problem

$$\text{Minimize}_{(d_x, d_y)} c_r(d_x, d_y) + \sum_{(i,j) \in S} c_{ij}(d_x, d_y),$$

with fixed  $S$ , is obtained from

$$D_{d_x} c_r(d_x, d_y) + D_{d_x} \sum_{(i,j) \in S} c_{ij}(d_x, d_y) = 0$$

$$D_{d_y} c_r(d_x, d_y) + D_{d_y} \sum_{(i,j) \in S} c_{ij}(d_x, d_y) = 0$$

Let  $A = \sum_{(i,j) \in S} Z_{ij}^{-1}$  and  $a = \sum_{(i,j) \in S} Z_{ij}^{-1} (\hat{x}_i - \hat{y}_j)$ . Then, the problem can be stated as

$$R^{-1} \begin{pmatrix} d_x \\ d_y \end{pmatrix} + \begin{pmatrix} +A & -A \\ -A & +A \end{pmatrix} \begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} a \\ -a \end{pmatrix} \tag{5}$$

so that  $(I \quad I) R^{-1} \begin{pmatrix} d_x \\ d_y \end{pmatrix} = 0$ . Several subcases are presented in the following subsections.

### 2.2.1 Case of a Full Rank Prior

The above equations assume that  $R$  is a full rank matrix. In this case, both  $d_x$  and  $d_y$  are observable and are uniquely determined as a solution of (5).

### 2.2.2 Case of a Bias Prior on Only One Sensor

Let us assume, for example, that  $R = \begin{pmatrix} R_x & 0 \\ 0 & 0 \end{pmatrix}$ . Then we replace  $R^{-1}$  with the Moore-Penrose inverse,  $R^+ = \begin{pmatrix} R_x^{-1} & 0 \\ 0 & 0 \end{pmatrix}$ . In this case, one has the relation  $(I \ I) R^+ \begin{pmatrix} d_x \\ d_y \end{pmatrix} = 0$  so that  $R_x^{-1} d_x = 0$  or  $d = -d_y$  and  $d_x = 0$  so that both are observable. In case  $R = \begin{pmatrix} 0 & 0 \\ 0 & R_y \end{pmatrix}$ , then  $d_y = 0$  and  $d = d_x$  are observable.

### 2.2.3 The General Case of a Rank Deficient Prior

We do not pursue this general case, but one can work out the details using the Moore-Penrose inverse of  $R$  or a singular value decomposition of  $R$ . A complete description can be found in the work of Danford, Kragel, and Poore.<sup>15</sup>

### 2.3 Case of No Prior

If there is no prior, then  $c_r(d_x, d_y)$  is not present in the formulation (1), and the equations defining  $d_x$  and  $d_y$  are

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} w \\ w \end{pmatrix} + \begin{pmatrix} \frac{1}{2} A^{-1} a \\ -\frac{1}{2} A^{-1} a \end{pmatrix}$$

where  $w$  is an arbitrary constant. In particular, the individual components  $d_x$  and  $d_y$  are not observable; however, the relative bias  $d = d_x - d_y = A^{-1} a$  is observable.

### 2.4 The Combined Problem

It is not necessary to treat the biases,  $d_x$  and  $d_y$ , separately; instead, we can combine them as  $d = d_x - d_y$ , which is already the case with the costs  $c_{ij}$ . The difficulty is with the prior, especially in the case that there is cross correlation between the two sensors.

**THEOREM 2.1.** *Let  $R$  be of full rank. At a solution of of the estimation problem (5),  $(I \ I) R^{-1} \begin{pmatrix} d_x \\ d_y \end{pmatrix} = 0$  so that  $E d_x + F d_y = 0$  where  $E = (I \ I) R^{-1} \begin{pmatrix} I \\ 0 \end{pmatrix}$  and  $F = (I \ I) R^{-1} \begin{pmatrix} 0 \\ I \end{pmatrix}$ . Then,*

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix}^T \begin{pmatrix} R_{xx} & R_{xy} \\ R_{yx} & R_{yy} \end{pmatrix}^{-1} \begin{pmatrix} d_x \\ d_y \end{pmatrix} = d^T (R_{xx} - R_{xy} - R_{yx} + R_{yy})^{-1} d.$$

if and only if

$$\begin{aligned} E d_x + F d_y &= 0 \\ d_x - d_y &= d. \end{aligned}$$

The proof of this result is presented in<sup>15</sup> and will not be presented here. Using the above theorem, we may solve a problem for the relative bias  $d$  and then recover the individual components  $d_x$  and  $d_y$ . We have not worked out the additional cases such as rank deficient priors here, but the linear algebra is straight forward provided one is familiar with the matrix inversion lemma and basic space decompositions involving range and null spaces.

The problem with a single  $d$  is that of equation (1) with  $c_r$  and  $c_{ij}$  replaced with

$$\begin{aligned} c_r(d) &= \frac{1}{2} d^T (R_{xx} - R_{xy} - R_{yx} + R_{yy})^{-1} d \\ c_{ij}(d) &= \frac{1}{2} (\hat{x}_i + d - \hat{y}_j)^T Z_{ij}^{-1} (\hat{x}_i + d - \hat{y}_j) \\ &\quad + \frac{1}{2} \ln[\beta_T^2 |2\pi Z_{ij}|] + \gamma_{ij} \text{ for } i \neq 0 \text{ and } j \neq 0. \end{aligned} \tag{6}$$

### 3. LOCAL SEARCH AND HEURISTICS

In this section we explain a local search algorithm and some different heuristics. We use this term “heuristic” to denote an algorithm that produces a good solution in a small amount of time, but provides no proof of optimality. Likewise, we use the term “local search” to denote an algorithm that starts from a candidate solution and then iteratively moves to a neighbor solution with the goal being to improve the solution quality. Indeed, the local search algorithm presented next is used as part of the heuristics and branch and bound algorithms.

#### 3.1 Local Search Algorithm

The local search algorithm starts with a given  $d$ , then iteratively solves for the assignment and updates the bias  $d$  until the assignments no longer change. Here is a more formal description.

- (a) Given a bias estimate  $d$ , compute a corresponding assignment  $S = \{(i, j) \mid x_{ij} = 1\}$  as a solution of the assignment problem (1), but with  $d$  fixed.
- (b) Given an assignment  $S$ , compute the bias by solving

$$\text{Minimize}_d \quad c_r(d) + \sum_{(i,j) \in S} c_{ij}(d).$$

- (c) Repeat a) and b) until the assignment no longer changes.

This is generally an expensive algorithm, but its use for a few instances is warranted.

#### 3.2 All-Pairs Heuristic

For each arc  $(i, j) \in A$ ,

- (a) Compute the solution  $d$  from  $D_d c_{ij}(d) = 0$ .
- (b) Given  $d$  from (a), apply the local search algorithm (3.1).

We have added a time improvement to this algorithm by

- (c) If during (b) an assignment  $S$  or displacement  $d$  has been computed in an earlier local search, then stop the local search.

The addition of part (c) has significantly reduced algorithm runtime in many cases.

#### 3.3 Centroid Heuristic

Compute the centroid of the states  $x_i$  ( $i = 1, \dots, m$ ) and the states  $y_j$  ( $j = 1, \dots, n$ ) to form centroids  $\bar{x}$  and  $\bar{y}$ , respectively. Set  $d = -\bar{x} + \bar{y}$ . Now start the local improvement algorithm with this displacement. We note in passing that Blackman and Popoli [3, Section 10.5] recommend the use of the componentwise median instead of a centroid.

### 3.4 Heuristics Based on K-Best Solutions

This class of heuristics starts with two initial preprocessing components: covariance inflation and cost shifting. To address bias uncertainty, the state covariance matrix is inflated by transforming the bias covariances at the sensor level to state space and adding them to the uncertainty in the state in what is usually called a consider analysis. (At the sensor level, one might use, e.g., the Schmidt-Kalman filter<sup>2</sup> to better develop these covariances instead of a consider analysis.) Another possibility is to inflate the reduced covariance  $(P_{ii} + Q_{jj} - S_{ij} - S_{ji})$  with  $(P_{ii} + Q_{jj} - S_{ij} - S_{ji} + D)$  in the combined problem (Section 2.4) with costs defined in equation (6). Here,  $D$  is a diagonal, positive definite matrix.

We also *shift* the cost until we have sufficient number of assignments. Here is one possibility. If  $\|d\|_{Z^{-1}}$  is the largest size to be considered where  $Z = (R_{xx} - R_{xy} - R_{yx} + R_{yy})$  from equation (6), then

$$\begin{aligned} \|x\|_{Z^{-1}} - \|d\|_{Z^{-1}} &\leq \|d - x\|_{Z^{-1}} \leq \sqrt{G} \\ \Rightarrow \|x\|_{Z^{-1}} &\leq \sqrt{G} + \|d\|_{Z^{-1}} \\ \Rightarrow \|x\|_{Z^{-1}}^2 &\leq G + 2\sqrt{G}\|d\|_{Z^{-1}} + \|x\|_{Z^{-1}}^2 \\ \Rightarrow \gamma_{ij} &\leftarrow \gamma_{ij} + 2\sqrt{G}\|d\|_{Z^{-1}} + \|d\|_{Z^{-1}}^2 \end{aligned}$$

where the value of  $G$  is a confidence bound for a chi-squared distribution with degrees of freedom equal to the dimension of the track state. For example, with a six dimensional state vector a 99.7% confidence bound is 19.8.

Assuming the covariance matrices have been inflated and the costs have been shifted, the k-best solutions are computed with  $d = 0$ . The arcs in the  $k^{th}$  best solution are denoted by  $S_k$ . Given this notation, one can generate several heuristics explained as follows.

**K-Best Intersection Heuristic.** For  $\ell = 1, \dots, k$  solve the least squares problem

$$\text{Minimize}_d \quad c_r(d) + \sum_{(i,j) \in \cap_{p=1}^{\ell} S_p} c_{ij}(d)$$

for a  $d_{\ell}$ . Note that the number of  $d$ 's generated is  $k$ . Next one should use the local search algorithm in Subsection 3.1 to improve the bias and assignment starting from each  $d_{\ell}$  ( $\ell = 1, \dots, k$ ).

**K-Best Union Heuristic.** Solve the least squares problem

$$\text{Minimize}_d \quad \sum_{\ell=1}^k \left[ c_r(d) + \sum_{(i,j) \in S_{\ell}} c_{ij}(d) \right]$$

for a  $d$ . The above local improvement algorithm is then employed to improve this solution.

**K-Best All-Pairs Heuristic.** For each arc  $(i, j) \in \cup_{\ell=1}^k S_{\ell}$  solve  $D_d c_{ij}(d) = 0$ . The number of  $d$ 's generated is provably  $O(K \text{Minimum}\{m, n\})$ , but is most often approximately  $\text{Minimum}\{m, n\} + K$ . The local improvement algorithm is employed to improve each of these  $d$ 's.

**K-Best Heuristic.** An algorithm that is competitive with the All-Pairs Heuristic in quality but superior in runtime is to first inflate covariances and use adaptive cost shifting to generate a good initial assignment. Then, we combine the K-Best Intersection Heuristic and K-Best All-Pairs Heuristic.

## 4. A BRANCH AND BOUND FRAMEWORK

The branch and bound method starts by considering the original problem with the complete feasible region, which is called the root problem. The lower-bounding and upper-bounding procedures are applied to the root problem. If the bounds match, then an optimal solution has been found and the procedure terminates; otherwise, the feasible region is divided into two or more regions, each a strict subregion of the original, which together cover the whole feasible region. These subproblems become children of the root search node, and, ideally, partition the feasible region. The algorithm is applied recursively to the subproblems, generating a tree of subproblems. If an optimal solution is found to a subproblem, it is a feasible solution to the full problem, but not necessarily globally optimal. Since it is feasible, it can be used to prune the rest of the tree: if the lower bound for a node exceeds the best known feasible solution, no globally optimal solution can exist in the subspace of the feasible region represented by the node. Therefore, the node can be removed from consideration. The search proceeds until all nodes have been solved or pruned, or until some specified threshold is met between the best solution found and the lower bounds on all unsolved subproblems. The technical details of this procedure are developed for the problem (1) in the following sections.

### 4.1 A Generic Branch and Bound Algorithm

The branch and bound procedure described above will be used to solve the optimization problem

$$\begin{aligned} & \text{Minimize } f(d, x) \\ & \text{Subject to } x \in F \\ & \quad d \in \mathbb{R}^n \end{aligned}$$

where  $F$  is a finite set. The variable  $d$  is a continuous variable that can either be  $d = d_x - d_y$  or  $d = \begin{pmatrix} d_x \\ d_y \end{pmatrix}$  discussed in Section 2, or possibly a much larger vector for the  $N$ -source pattern matching problem. The branch and bound procedure uses an acyclic graph known as a branch and bound tree that forms progressively finer partitions of the finite set  $F$  defined by the constraints and integer variables in equation (1). The nodes in the tree correspond to a collection  $\mathcal{F}$  of subsets of  $F$ . Also, the algorithm maintains a node list called ‘‘OPEN’’ and an upper bound ‘‘UPPER,’’ which is the minimal cost of all feasible solutions found so far. OPEN is initialized with the node  $F$  along with corresponding information such as the lower bound, while UPPER is set equal to the cost of a feasible solution  $(\bar{d}, \bar{x})$  found by a heuristic such as one of those noted in Section 3.

### Salient Features of the Branch and Bound Algorithm

**Step 1.** Initialization: Compute a global lower bound  $LB(F)$ , feasible solution  $(\bar{d}, \bar{x})$ , and an upper bound  $UB(F) = f(\bar{d}, \bar{x})$ . If  $LB(F) = UB(F)$ , stop for the optimal solution  $(\bar{d}, \bar{x})$  has been found; otherwise, continue.  $OPEN = \{(F, LB(F))\}$  and  $UPPER = UB(F)$ .

**Step 2.** Remove a node  $N$  from OPEN.

**Step 3.** If  $LB(N) \geq UPPER$ , the node  $N$  is said to be fathomed and can be deleted from further consideration; otherwise, find a global solution  $(\bar{d}, \bar{x})$  for the node  $N$  or partition  $N$  into  $\{N_1, \dots, N_k\}$  nodes. In the former case, if  $f(\bar{d}, \bar{x}) < UPPER$ , set  $UPPER = f(\bar{d}, \bar{x})$ , mark  $(\bar{d}, \bar{x})$  as the best solution so far, and go to Step 4. In the latter case, for each child  $N_i$  of  $N$ , do the following

**3a.** Compute a lower bound  $LB(N_i)$ . If  $LB(N_i) < UPPER$ , place the node  $N_i$  in OPEN:

$$OPEN = OPEN \cup \{(N_i, LB(N_i))\}; \tag{7}$$

Otherwise,  $N_i$  is said to be fathomed.

**3b.** Compute a feasible solution  $(\bar{d}, \bar{x})$  and an upper bound  $UB(N_i) = f(\bar{d}, \bar{x})$ . If  $UB(N_i) = f(\bar{d}, \bar{x}) < UPPER$ , set  $UPPER = f(\bar{d}, \bar{x})$  and mark  $(\bar{d}, \bar{x})$  as the best solution so far.

**Step 4.** Termination Test: If OPEN is nonempty, go to Step 2; otherwise, terminate as the best solution found thus far is optimal.

In Step 1 above, the case  $LB(F) = UB(F)$  will almost never occur due to the sensor and track state uncertainty. In the following, we address each of these steps in the above generic algorithm.

## 4.2 The Root Node

The root node in the branch and bound tree is defined by the original problem (1). For this node, one must supply a global lower bound, an initial feasible solution and its corresponding upper bound, along with a partition of the problem into disjoint components.

### 4.2.1 A Global Lower Bound for the Root Node

For branch and bound, the lower bound that one must compute at each node in the branch and bound tree must be a global one. To achieve this, one often relaxes the discrete variables to continuous ones and develops convex lower estimates of the objective functions and constraints that facilitate the computation of this global lower bound. An example of this is the  $\alpha$ -branch and bound method of Adjiman, Androulakis, and Floudas,<sup>16</sup> which can be applied to the current problem.

The branch and bound framework developed herein does not follow these approaches for MINLP problems. The constraints are already convex in that they are affine. The objective function is nonconvex; however, we can develop a convex (affine) lower approximation to the objective function by using

$$\begin{aligned} b_{ij} &= \frac{1}{2} \ln[\beta_T^2 |2\pi Z_{ij}|] + \gamma_{ij} \text{ for } i, j \neq 0, \\ b_{ij} &= 0 \text{ for } i = 0 \text{ or } j = 0, \end{aligned} \quad (8)$$

which are independent of the displacement  $d$ . Furthermore,  $b_{ij} \leq c_{ij}(d)$  for all  $i, j$ , thereby providing a basis for the global lower bound needed in the branch and bound procedure. We can make effective use of an assignment solver, so that there is no need to relax the discrete variables.

### 4.2.2 The Initial Feasible Solution

The initial feasible solution that serves as an upper bound on the optimal solution can be obtained by any of the heuristics in Section 3. Such a solution produces a bias estimate  $\bar{d}$  and an assignment  $S = \{(i, j) \in \mathcal{A} \mid \bar{x}_{ij} = 1\}$ .

## 4.3 A Descendant Node

In this section, a descendant node in the branch and bound tree, the optimization problem, a global lower bound for the node, and the construction of a feasible solution are formulated.

A node in the branch and bound tree is defined by arcs  $(k, \ell) \in A$  that are in the assignment,  $S = \{(k, \ell) \neq (0, 0) \mid x_{k\ell} = 1\}$ , arcs declared not to be in any assignment for that node,  $D = \{(k, \ell) \neq (0, 0) \mid x_{k\ell} = 0\}$ , and arcs that are free to be assigned,  $A - S - D$ . For the root node,  $S = D = \emptyset$  and for a leaf node,  $A = S \cup D \cup \{(0, 0)\}$ .

### 4.3.1 The Optimization Problem at a Descendant Node

Define

$$\begin{aligned} \tilde{A}(i) &= \{j \mid (i, j) \in A - S - D \text{ for some } j\}, \\ \tilde{B}(j) &= \{i \mid (i, j) \in A - S - D \text{ for some } i\}, \\ I &= \{i \mid \tilde{A}(i) \text{ contains at least one nonzero index}\}, \\ J &= \{j \mid \tilde{B}(j) \text{ contains at least one nonzero index}\}. \end{aligned}$$

Then, the optimization problem for each node has the form

$$\begin{aligned}
& \text{Minimize}_{(d,x)} \left[ c_r(d) + \sum_{(i,j) \in S} c_{ij}(d) + \sum_{(i,j) \in A-D-S} c_{ij}(d)x_{ij} \right] \\
& \text{Subject To: } \sum_{j \in \tilde{A}(0)} x_{0j} = |J|, \\
& \sum_{j \in \tilde{A}(i)} x_{ij} = 1 \quad (i \in I, i \neq 0), \\
& \sum_{i \in \tilde{B}(0)} x_{i0} = |I|, \\
& \sum_{i \in \tilde{B}(j)} x_{ij} = 1 \quad (j \in J, j \neq 0), \\
& x_{ij} \in \begin{cases} \{0, 1\} & \text{for } i \neq 0 \text{ or } j \neq 0, \\ \{0, 1, \dots, \text{Min}\{|I|, |J|\}\} & \text{for } i = 0 \text{ and } j = 0, \end{cases}
\end{aligned} \tag{9}$$

#### 4.3.2 A Lower Bound at a Descendant Node

A lower bound is obtained by replacing  $c_{ij}(d)$  in the final term in the objective function in equation (9) by  $b_{ij}$  from equation (8) for each  $(i, j) \in A-D-S$ . The objective function in (9) then decomposes into two components:

$$\text{Minimize}_d \left[ c_r(d) + \sum_{(i,j) \in S} c_{ij}(d) \right] + \text{Minimize}_x \left[ \sum_{(i,j) \in A-D-S} b_{ij}x_{ij} \right]$$

subject to the constraints in (9). The two optimization problems in this equation can be solved efficiently by starting with the corresponding solutions of the parent node.

#### 4.3.3 A Feasible Solution and Upper Bound for the Descendant Node

While there are several methods for computing an upper bound for a descendant node, here are a couple that we have found to work well. The first is to start with the solution  $d$  of

$$\text{Minimize}_d \left[ c_r(d) + \sum_{(i,j) \in S} c_{ij}(d) \right],$$

which is computed in the lower bound. The second would be to take the  $d$  from the parent feasible solution. For either case, the resulting  $d$  is used to obtain an assignment  $P = \{(i, j) \in A-S-D \mid x_{ij} = 1\}$ . This yields a  $(\bar{d}, \bar{x})$  with  $\bar{d} = d$  and  $\bar{x}_{ij} = 1$  if  $(i, j) \in S \cup P$ . The corresponding upper bound is given by  $c_r(\bar{d}) + \sum_{(i,j) \in S \cup P} c_{ij}(\bar{d})$ . This upper bound and corresponding solution  $(\bar{d}, \bar{x})$  can then be improved by using the local search algorithm discussed in Section 3.1 adapted to the problem (9).

#### 4.4 Partitioning a Node

Having explained the optimization problem, the lower bound, and the construction of a feasible solution that serves as an upper bound for the node, we now turn to the partitioning algorithm. One could use the partition developed by Murty<sup>17</sup> modified to the current inequality constrained problem. A second method of partitioning and the one used in this work is to partition a node based on the different ways a constraint can be satisfied. For example, given a constraint  $\sum_{j \in \tilde{B}(i)} x_{ij} = 1$  for some  $i \in I$  we could enumerate the  $j \in \tilde{B}(i)$  by  $\{j_0, j_1, \dots, j_r\}$  with  $j_0 = 0$ . For each of these values  $j_p$  we create a child node, in which  $x_{ij_p} = 1$ ,  $x_{ij} = 0$  for  $j \neq j_p$ , and  $x_{kj_p} = 0$  for  $k \neq i$ . Since  $\sum_{j \in \tilde{B}(i)} x_{ij} = 1$ , the union of the corresponding nodes is the parent, and the intersection of any two of these nodes is the empty set. Therefore, the solution spaces of these nodes form a partition of the solution space of the parent node.

## 4.5 Search Procedure for Node Selection

The choice of search procedure for selecting the next node to expand in Step 2 in the branch and bound procedure in Section 4.1 can significantly impact the speed and storage in the procedure. Breadth-first search (BFS), which is normally implemented as a queue, depth-first search (DFS), implemented as a stack, and best-first search, implemented as a priority queue, are often used for branch and bound procedures. Since there are different ways to decide the “estimated best”, there are variants of best-first search: uniform-cost search (estimated best is the least cost so far), greedy search (least estimated cost to goal), A\* (cost so far plus estimated cost to goal), and many refinements of those. The many options can be found in the book by Russell and Norvig [18, Chapter 4]. For the best first search, we could use the lower bound or the upper bound. In the following, we discuss the best first search based on the lower bound, which is an A\*-search procedure, and the depth-first search with node ordering, i.e., A\* per level.

### 4.5.1 A\*-Search

Given a node  $N$  defined by the arc sets  $S$  and  $D$ , the lower bound is defined by

$$\text{Minimize}_d \left[ c_r(d) + \sum_{(i,j) \in S} c_{ij}(d) \right] + \text{Minimize}_x \left[ \sum_{(i,j) \in A-D-S} b_{ij}x_{ij} \right],$$

wherein, one can identify  $\text{Minimize}_d \left[ c_r(d) + \sum_{(i,j) \in S} c_{ij}(d) \right]$  as the backward cost from the current node to the root node and  $\text{Minimize}_x \left[ \sum_{(i,j) \in A-D-S} b_{ij}x_{ij} \right]$  as the underestimate of the forward cost to any leaf node from node  $N$ . Thus, this could be used for A\* search. The lower bound as a heuristic is reasonably cheap to compute given similar information from the parent node. One can show that this overall estimate always underestimates the objective function.

Such a heuristic is said to be *admissible*. One can also show that the estimate is nondecreasing from parent to child. This property is known as *consistency*. When these two properties hold, A\*-search will always find the optimal solution on any graph as the first goal node selected for expansion. However, since the graph being searched (the branch and bound tree) contains no cycles, admissibility is sufficient for this case.

To explain, we first note that at a leaf node the lower bound is equal to the cost of the single feasible solution at that node. Now, suppose that the next node selected from the priority queue (the one with the lowest lower bound) is a leaf node. This means that the lower bound on any other unevaluated subproblem is greater than the cost of the solution at the leaf node. Since no solution can have a cost less than the lower bound of the node in which it lies, the solution to that leaf node is the solution to the global problem. This guarantees that the first leaf node selected by the priority queue in an A\*-search is the optimal solution.

Another useful property of A\*-search is that it is optimally efficient. This means that it expands fewer nodes than any different search procedure that guarantees the optimal solution. To demonstrate this, we note from the reasoning above that A\*-search never expands a node that has a lower bound greater than the optimal solution.<sup>19</sup> If any different search procedure chooses not to expand a node that has a lower bound less than the optimal solution, then it cannot know that the solution it provides is optimal, because in theory there could be a better solution within that node, if it had expanded it. Any search procedure that is optimal must therefore search every node that has a lower bound less than the optimal solution. Since A\* expands *only* those nodes, it therefore expands the fewest possible nodes of any optimal search procedure.

The drawback of A\*-search is that it requires memory that grows exponentially with the problem size. This also affects the speed due to the overhead of maintaining a priority queue. Several search procedures such as depth-first, iterative deepening, and recursive best-first search use memory more efficiently in that their memory usage grows only linearly with the problem size. These methods were compared for a variety of problem types in the work of Zhang and Korf.<sup>20</sup>

### 4.5.2 Depth-First Search

In a depth-first branch and bound algorithm, the node with most fixed arcs, i.e.,  $(i, j) \in S \cup D$ , is chosen to explore next. Since many nodes may have the same number of constraints at the same level in the branch and bound tree, ties are broken by selecting the node with the lowest lower bound. This tie breaking rule, which is essentially A\* per level, is called *node ordering*. A depth-first search requires memory that grows linearly with the problem size. Thus, its memory requirements are small compared to A\*-search and it can be used on much larger problems without running out of space or spending a great deal of time maintaining a queue.

As an exhaustive search procedure, a depth-first search with node ordering will in general evaluate more nodes than A\*-search. However, within the branch and bound framework no search procedure expands any nodes with a lower bound greater than their *current* upper bound. Since this may not be the optimal cost, they can evaluate some additional nodes only if those nodes' lower bounds lie between their current best and the actual best. Thus, if the upper bound from the heuristic provides the optimal cost, then a depth-first search (or any optimal search procedure) will evaluate exactly the same nodes as an A\*-search evaluates, but in a different order.

If the initial upper bound is very close to the optimal solution, then it is reasonable to think that very few nodes will have a lower bound between the upper bound and the optimal cost. Since the optimal solution may be found (thus lowering the upper bound to the optimal cost) before any such nodes are expanded, a depth-first search may still not expand any additional nodes. The closer the starting upper bound is to the optimal cost, the fewer extra nodes a depth-first search might need to evaluate compared with A\*-search.

## 4.6 Computational Efficiency

Several auxiliary procedures can be used to improve the efficiency of the branch and bound procedure. We discuss four of these in this section.

### 4.6.1 Partitioning the Assignment Problem into Disjoint Connected Components

Decomposition of the assignment problem (1) is accomplished by determining the connected components of the associated bipartite graph defined by the constraints. To explain, let

$$\mathcal{X} = \{x_{ij} \mid (i, j) \in A\}$$

denote the set of assignable variables. Define an undirected graph  $\mathcal{G} = (\mathcal{N}, \mathcal{A})$  where the set of arcs,  $\mathcal{A} = \{(i, j) \mid (i, j) \in A\}$ . Notice that the nodes corresponding to the zero index have been eliminated. Connected components of the graph are then easily found by constructing a spanning forest via a depth-first search. A detailed algorithm can be found in Section 5.2 of the book by Aho, Hopcroft and Ullman.<sup>21</sup> This results in independent smaller problems.

### 4.6.2 The Solution of the Lower Bound Assignment Problem

A child node is obtained from the parent by adding arcs to  $S$  and  $D$ . For the partitioning presented above, one removes arcs and places one in  $S$  and the remainder in  $D$ . The optimal solution of the child assignment problem involving the  $b_{ij}$ 's, can be achieved by one augmentation of the shortest path algorithm.

### 4.6.3 The Bias Estimation Problem

Note that the problem  $\text{Minimize } {}_d c_r(d) + \sum_{(i,j) \in S} c_{ij}(d)$  is a linear least squares problem and is obtained by adding costs  $c_{k\ell}(d)$  to a corresponding sum of costs in the parent. Thus, one can use recursive least squares to update  $d$  efficiently. Having said this, we find that the most expensive part of the algorithm is that of evaluating the costs  $c_{ij}(d)$  given the different values of  $d$  that arise in the algorithm.

#### 4.6.4 Gating Arcs in a Node

It is possible to gate the arcs in the assignment problem for each node  $N$  by first observing that costs in an optimal solution must satisfy  $c_{ij}(d) \leq 0$ . Recall the optimization problem has the objective function Minimize  $_{(d,x)} \left[ c_r(d) + \sum_{(i,j) \in S} c_{ij}(d) + \sum_{(i,j) \in A-D-S} c_{ij}(d)x_{ij} \right]$ . The costs  $c_{ij}(d)$  ( $(i,j) \in S$ ) define a region in the bias space, say  $R_S$  in which  $d$  must lie. Thus, any arc  $(i,j) \in A-D-S$  for which  $\{d \mid c_{ij}(d) \leq 0\} \cap R_S = \emptyset$  can be ruled out from further consideration in the optimization problem. While this region is too complex to implement, there is a less stringent gate that can be efficiently implemented. The idea is to replace each  $\{d \mid c_{ij}(d) \leq 0\}$  with the smallest cuboid containing this set. To explain, let  $C \in \Re^{p \times p}$  be positive definite. Then

$$\{\epsilon \mid \epsilon^T C^{-1} \epsilon \leq G\} \subset \{\epsilon \mid |\epsilon_k| \leq \sqrt{GC_{kk}} \text{ for } k = 1, \dots, p\}$$

where  $C_{kk}$  is the  $k^{\text{th}}$  diagonal element in  $C$ . This fact motivates the definition of the cuboid

$$B_{ij}(d) = \left\{ d \mid |(x_i - (d + y_j))_k| \leq \sqrt{[-\ln[\beta_T^2 |2\pi Z_{ij}|] - 2\gamma_{ij}](Z_{ij})_{kk}} \right\},$$

where  $Z_{ij} = P_{ii} + Q_{jj} - S_{ij} - S_{ji}$ . Then any arc  $(i,j)$  for which  $B_{ij}(d) \cap B_S = \emptyset$  where  $B_S = \cap_{(i,j) \in S} B_{ij}(d)$  cannot be in the optimal solution with those already in  $S$ .

#### 4.6.5 $\epsilon$ -Optimality

In the branch and bound algorithm, a node  $N_i$  is placed in the list OPEN if  $LB(N_i) < UPPER$ . The  $\epsilon$  algorithm generally replaces this by  $LB(N_i) < UPPER(1 - \epsilon_1) - \epsilon_2$ , resulting in a final result that is optimal only to within  $O(\epsilon_1, \epsilon_2)$ .

#### 4.7 K-Best Solutions

The branch and bound procedure supports the construction of the k-best solutions with a slight modification. There are two possible approaches here. First, the heuristic can be used to generate  $k$  solutions. We then maintain a list of  $k$  solutions throughout the branch and bound procedure. In order to fathom nodes, the upper bound ‘‘UPPER’’ in the branch and bound algorithm in Section 4.1 is replaced by the value of the  $k^{\text{th}}$ -solution found thus far as opposed to the best solution. Alternately, if we do not start with k-solutions, then we would not fathom a node until k-solutions (upper bounds) are computed. Then, the above scheme is appropriate.

### 5. STAND-ALONE A\*-SEARCH

The A\*-search procedure embedded in the the above branch and bound framework can stand alone and yields an optimal solution. Of course, this best first search based on the lower bound still requires the partitioning algorithm in Section 4.4 to create a tree structure, but it does not require that one compute an upper bound at each node. The GNPL algorithm due to Levedahl<sup>4,5</sup> is an A\*-search in this sense, since the costs are shifted to be nonnegative in which case his search procedure is also based on a lower bound. A noteworthy observation for A\*-search is that the first k-leaf nodes explored in the A\*-search are the k-best solutions. What is more, this search can be made even more efficient through the use of gating in Section 4.6.4. Additionally, the upper bound obtained from the heuristics in Section 3 can be used to fathom nodes in the A\* tree.

### 6. CONCLUSIONS

The philosophy behind the development of a class of algorithms for the joint MAP estimation of biases and data association in this work has been to combine a good initial heuristic with local search followed by a full or partial branch and bound as time allows. If the branch and bound procedure is stopped prior to completion, one can use the local search procedure to obtain local minimizers for the nodes still in the list of nodes to explore. We have also addressed the linear least squares estimation problem associated with an assignment; however, the general framework applies equally well to linear and nonlinear cost functions.

We have explored two different heuristics in Section 3, namely, the “all-pairs” and the k-best heuristics. Computational experience<sup>14</sup> shows that the “all-pairs” algorithm produces very good or near-optimal solutions, but its runtime is unsatisfactory for larger problems. The k-best heuristic on the other hand has been designed to be competitive with the “all-pairs” algorithm, but with a superior runtime performance. This heuristic also provides any branch and bound algorithm with an exceptional initial upper bound and, thus, assists in the early pruning of branches that might otherwise need to be explored.

As presented in Section 4, there are many choices in the design of a branch and bound algorithm. We have pursued the design of the A\* and depth-first search procedures. A\*-search is optimal in that it explores the smallest number of nodes among any search procedure that uses the same heuristic. The very first leaf node that A\* explores is an optimal solution. If A\* is allowed to continue, then it will compute a guaranteed K-best bias-assignment pairs. The deficiency of A\* is that the memory required to store nodes can grow exponentially in the size of the problem. Depth-first search has memory that grows linearly in the size of the problem; however, it can explore substantially more nodes than A\*. If the initial upper bound is optimal, the two algorithms explore exactly same number of nodes. One would conjecture that if the initial upper bound is near optimal, then the number of nodes visited when using DFS with node ordering would be near that of A\*-search. Indeed, the simulations<sup>14</sup> support this conjecture.

As noted in Section 4.7, the computation of the k-best solutions of the bias-association fits nicely within the branch and bound framework. Also, the first k-leaf nodes in a stand-alone A\*-search algorithm in Section 5 are the k-best solutions.

Computationally, we have presented several techniques for enhancing the performance of the branch and bound procedure including partitioning, gating, and  $\epsilon$ -optimality. Computational experience with these algorithms is presented in a part two of this paper.<sup>14</sup>

Finally, though we formulated the bias as a translation between the states, a better model might be achieved by going back to the radar itself and modeling the measurement biases and navigation errors. Future work will explore these avenues as well as the improved efficiency and multidimensional assignment (see the Appendix A) versions of the problem.<sup>13</sup>

**Acknowledgements.** This work was supported by the Air Force Office of Scientific Research through AFOSR Contract Number FA9550-04-1-0222 and by the Office of Naval Research through ONR Contract Number N00014-05-C-0413.

## APPENDIX A. THE MAP ASSIGNMENT PROBLEM

In this section, we derive the two-sensor MAP assignment problem including absolute bias estimation as a special case of the multisensor assignment problem as derived by Kragel, et al.<sup>13</sup>

### A.1 Problem Formulation

Consider  $n \geq 0$  objects moving independently within a surveillance region  $\mathcal{X}$  common to  $M \geq 2$  sensors, where neither the number of objects, nor their true positions  $x_i$ ,  $i = 1, \dots, n$ , is known *a priori*. Each sensor detects and tracks a subset of the objects; that is, we consider the possibility of missed detections\*. Periodically, the sensors report<sup>†</sup> the estimated track states (kinematic state and possibly feature and/or attribute data) to a central node for fusion. The sensor track reports take the form

$$\begin{aligned} \hat{x}_{i_1} &= x_{i_1} - d_1 + \nu_{i_1}, & i_1 &= 1, \dots, n_1, \\ \hat{x}_{i_2} &= x_{i_2} - d_2 + \nu_{i_2}, & i_2 &= 1, \dots, n_2, \\ & \vdots & & \end{aligned}$$

---

\*Since the sensors are assumed to report only established tracks, we do not consider false alarms.

<sup>†</sup>The reports are presumed to be time-aligned, either because the sensors report all tracks simultaneously, or by propagating the tracks to a common time.

$$\hat{x}_{i_M} = x_{i_M} - d_M + \nu_{i_M}, \quad i_M = 1, \dots, n_M,$$

where for sensor  $m$ ,  $m \in \{1, \dots, M\}$ ,  $n_m$  denotes the number of tracks reported by the sensor,  $\hat{x}_{i_m}$  denotes the estimated state of sensor track  $i_m$ ,  $i_m = 1, \dots, n_m$ ,  $x_{i_m}$  is the corresponding true, but unknown, state of the object corresponding to sensor track  $\hat{x}_{i_m}$ ,  $d_m$  is an unknown translational bias error for sensor  $m$ , and  $\nu_{i_m}$  denotes the random estimation error, which is assumed to be a zero-mean Gaussian with covariance  $P_{i_m}$ .

The problem to be addressed is that of determining which, if any, of the sensor reports emanate from common objects, while concurrently computing an estimate of the sensor biases  $d = (d_1, d_2, \dots, d_M)^T$ . Generalizing the work by Levedahl,<sup>4,5</sup> where only the *relative* bias between sensors was considered, prior bias estimates  $\hat{d}$  can be included by utilizing the covariance  $R = E[(d - \hat{d})(d - \hat{d})^T]$ , which is assumed to be known.

To formalize the subsequent discussion, we follow Poore<sup>22</sup> and let  $\mathcal{Z}(m) = \{\hat{x}_{i_m}\}_{i_m=1}^{n_m}$  denote the reports from sensor  $m$ ,  $m = 1, \dots, M$ , while  $\mathcal{Z} = \{\mathcal{Z}(1), \dots, \mathcal{Z}(M)\}$  denotes the set of all sensor reports. Similarly,  $\mathcal{I} = \{\mathcal{I}(1), \dots, \mathcal{I}(M)\}$ , with  $\mathcal{I}(m) = \{1, \dots, n_m\}$ ,  $m = 1, \dots, M$ , denotes the indices assigned to the reports. A partition  $H$  of  $\mathcal{I}$  and the collection  $\mathcal{H}$  of all such partitions are defined by

$$H = \{h_1, \dots, h_{n(H)} \mid h_k \subset \mathcal{I}, h_k \neq \emptyset, k = 1, \dots, n(H)\}, \quad (10)$$

$$|h_k \cap \mathcal{I}(m)| \leq 1, \quad k = 1, \dots, n(H), \quad m = 1, \dots, M \quad (11)$$

$$h_k \cap h_j = \emptyset, \quad \text{for } k \neq j, \quad (12)$$

$$\mathcal{I} = \cup_{k=1}^{n(H)} h_k, \quad (13)$$

$$\mathcal{H} = \{H \mid H \text{ satisfies (10) - (13)}\}, \quad (14)$$

where the number  $n(H)$  of subsets (or tracks) comprising a particular partition or hypothesis will vary from one hypothesis to another. We shall refer to the partitions  $H$  as *track-to-track data association hypotheses* and the subsets  $h_k \subset \mathcal{I}$  as *(system) tracks*. Conditions (12) and (13) ensure that each sensor report is assigned to one and only one track, while condition (11) ensures that no track contains more than one report from any particular sensor; thus, every sensor report is assigned uniquely to a track and no two reports from the same sensor are assigned to the same track. In this manner, each partition (or hypothesis)  $H$  induces a partition of the data  $\mathcal{Z}$  via

$$\mathcal{Z}_H = \{\mathcal{Z}_{h_1}, \dots, \mathcal{Z}_{h_{n(H)}}\}, \quad \mathcal{Z}_{h_k} = \{\hat{x}_{h_{kj}}\}_{j=1}^{n(h_k)}.$$

Clearly,  $\mathcal{Z}_{h_k} \cap \mathcal{Z}_{h_j} = \emptyset$  for  $k \neq j$  and  $\mathcal{Z} = \cup_{k=1}^{n(H)} \mathcal{Z}_{h_k}$ .

Note that each track  $h_k = \{h_{kj}\}_{j=1}^{n(h_k)}$ , where  $n(h_k) = |h_k|$ , is composed of sensor track indices; that is,  $h_{kj} = i_m$ , where  $i_m$  implies both a sensor  $m$  and a sensor report index  $i_m \in \{1, \dots, n_m\}$ . Through a slight abuse of notation<sup>‡</sup>, this allows us to conveniently reference the sensors and sensor reports comprising each hypothesized track by appending the track notation as a subscript to the component of interest; thus,  $d_{h_{kj}}$  denotes the sensor making report  $h_{kj}$ , while  $d_{h_k}$  denotes the vector, ordered by sensor index, of all sensors reporting on track  $h_k$ . For example, if  $h_k$  is composed of, say, sensor report 3 from sensor 2 and sensor report 1 from sensor 5, then  $d_{h_k} = (b_2, b_5)^T$ . Similarly, we use  $\hat{x}_{h_k}$  to refer to the vector of sensor reports, ordered by sensor index, postulated by  $h_k$  to emanate from a common object.

In order to avoid dimensionality issues, it is useful to formulate the data association problem in terms of a dimensionless likelihood ratio; thus, based on the statistics of the reports from the sensors, we seek the maximum *a posteriori* (MAP) assignment (or hypothesis)  $H^* \in \mathcal{H}$ , and bias  $d^*$  for which

$$\Lambda(H^*, d^* | \mathcal{Z}) = \max_{H, d} \Lambda(H, d | \mathcal{Z}), \quad (15)$$

where

$$\Lambda(H, d | \mathcal{Z}) = \frac{\Pr(H, d | \mathcal{Z})}{\Pr(H_0, d | \mathcal{Z})}, \quad (16)$$

<sup>‡</sup>The abuse of notation arises from allowing  $h_{kj}$  to reference a sensor in some contexts and a sensor track in others. Which is meant should always be clear from the context.

$H_0$  is the null hypothesis postulating that none of the reports from any of the sensors emanate from a common object, and  $\bar{\text{Pr}}(\cdot, \cdot | \cdot)$  is a reference likelihood in which no prior bias estimate exists. For  $\bar{\text{Pr}}(\cdot, \cdot | \cdot)$ , the bias is *a priori* assumed to be uniformly distributed in some volume  $\zeta_d$  with the same dimension as the bias.

## A.2 MAP Derivation

To formulate (15) as an assignment problem, we use the following theorem and corollary from Kragel, et al.<sup>13</sup>

**THEOREM A.1.** *Assume for  $M$  sensors with a common surveillance region  $\mathcal{X}$  that the following conditions hold: a) the unknown number of objects in the surveillance region  $\mathcal{X}$  is Poisson distributed with expected value  $\hat{\eta} = E[n]$  and the objects are moving independently within  $\mathcal{X}$ ; b) the probability of detection is independent of target position; c) the prior density of the true target location is uniform throughout the surveillance region; d) the joint prior density of the sensor bias is a zero mean Gaussian; and e) the sensor bias is independent of the true object position. Then*

$$\Lambda(H, d | \mathcal{Z}) = \frac{1}{|2\pi R|^{1/2} \zeta_d^M} \exp\left(-\frac{1}{2} d^T R^{-1} d\right) \cdot \prod_{h_k \in H} \frac{|2\pi (B_{h_k}^T P_{h_k}^{-1} B_{h_k})^{-1}|^{1/2} \exp\left(-\frac{1}{2} \hat{z}_{h_k}^T \left(P_{h_k}^{-1} - P_{h_k}^{-1} B_{h_k} (B_{h_k}^T P_{h_k}^{-1} B_{h_k})^{-1} B_{h_k}^T P_{h_k}^{-1}\right) \hat{z}_{h_k}\right)}{|2\pi P_{h_k}|^{1/2} \beta_T^{|h_k|-1} \prod_{m=1}^M (1 - P_D^m)^{|h_k|-1}}, \quad (17)$$

where  $R = E[dd^T]$  is the joint covariance of the aggregate bias vector  $d = (d_1, \dots, d_M)^T$ ,  $P_{h_k}$  is the joint covariance of the sensor reports comprising track  $h_k$ ,  $B_{h_k} = [I, \dots, I]^T \in \mathbb{R}^{p \times n(h_k), m}$ , where  $I \in \mathbb{R}^{p, p}$  denotes the identity matrix and  $p$  is the dimension of the object state space,  $\beta_T = \hat{\eta}/V$  is the (constant) target density in  $\mathcal{X}$ , where  $V$  is the volume of  $\mathcal{X}$ ,  $P_D^m$  is the probability of detection for sensor  $m$ , and  $\hat{z}_{h_k} = \hat{x}_{h_k} + d_{h_k}$ .

We emphasize that (17) is a dimensionless likelihood. It is also worth noting that the matrices  $B_{h_k}$ ,  $h_k \in H$  arise from the fact that sensor measurements hypothesized to emanate from a common object are dependent samples. Taking this dependency into account, our formulation explicitly accounts for cross correlation between track reports on common objects, which may arise due to common process noise or common priors in the sensor estimation filters.

For the special case of two sensors, we can simplify notation by dropping the sensor subscripts on the sensor report indices and simply using  $i$  to index the sensor 1 reports, while  $j$  indexes the sensor 2 reports; thus we consider two sets of sensor reports

$$\begin{aligned} \hat{x}_i &= x_i - d_1 + \mu_i, & i &= 1, \dots, n_1, \\ \hat{y}_j &= y_j - d_2 + \nu_j, & j &= 1, \dots, n_2, \end{aligned}$$

where  $\mu_i$  and  $\nu_j$  are zero-mean Gaussians with covariances  $P_{ii}$  and  $Q_{jj}$ , respectively. In this manner, each nontrivial track (those containing more than one sensor report) is composed of a single index pair  $(i, j)$ . As shown in Kragel, et al.,<sup>13</sup> with these conventions, the following corollary follows from Theorem A.1.

**COROLLARY 1.** *For the two-sensor case, (17) reduces to*

$$\Lambda(H, d | \mathcal{Z}) = \frac{1}{|2\pi R|^{1/2} \zeta_d^2} \exp\left(-\frac{1}{2} d^T R^{-1} d\right) \cdot \prod_{(i,j) \in H} \frac{1}{|2\pi Z_{ij}|^{1/2} \beta_T (1 - P_D^1) (1 - P_D^2)} \exp\left(-\frac{1}{2} (\hat{z}_i^1 - \hat{z}_j^2)^T Z_{ij}^{-1} (\hat{z}_i^1 - \hat{z}_j^2)\right), \quad (18)$$

where  $\hat{z}_i^1 = \hat{x}_i + d_1$ , and  $\hat{z}_j^2 = \hat{y}_j + d_2$ ,  $d = (d_1, d_2)^T$ ,  $R = E[dd^T]$  and

$$Z_{ij} = (P_{ii} + Q_{jj} - S_{ij} - S_{ji});$$

with  $S_{ij} = S_{ji}^T = E[\mu_i \nu_j^T]$  denoting the cross-covariance between report  $i$  from sensor 1 and report  $j$  from sensor 2.

Note that the product in (18) explicitly enumerates only nontrivial tracks, since the likelihoods in (17) for tracks consisting of only one sensor report evaluate to unity.

### A.3 Assignment Problem Formulation

Now that we have a useful expression (18) for the likelihood ratio  $\Lambda(H, d|\mathcal{Z})$ , we are in a position to formulate the two-dimensional data association problem as an optimization problem over  $\mathcal{H} \times \mathbb{R}^{2p}$ , where  $p$  is the dimension of the bias. To this end, let  $S$  denote the set of nontrivial tracks (index pairs)  $(i, j)$  in the hypothesis  $H \in \mathcal{H}$ . In the assignment problem context, elements of  $S$  are referred to as *arcs* and the set of arcs corresponding to a given hypothesis as an *assignment*. Note that in this sense  $H \leftrightarrow S$ , with the notation change made to reflect the fact that we are now working in an optimization, rather than a probabilistic, context.

We begin by transforming the maximization problem (15) with the multiplicative objective function (18) into an equivalent minimization problem with a more convenient additive objective function

$$-\ln(\Pr(H, d|\mathcal{Z})) = c_r(d) + \sum_{(i,j) \in S} c_{ij}(d) + \gamma_r, \quad (19)$$

where

$$\begin{aligned} c_r(d) &= \frac{1}{2} d^T R^{-1} d, \\ c_{ij}(d) &= \frac{1}{2} (\hat{z}_i^1 - \hat{z}_j^2)^T Z_{ij}^{-1} (\hat{z}_i^1 - \hat{z}_j^2) + \frac{1}{2} \ln |2\pi Z_{ij}| + \gamma_{ij}, \\ \gamma_{ij} &= \ln(\beta_T (1 - P_D^1) (1 - P_D^2)), \\ \gamma_r &= \frac{1}{2} \ln |2\pi R| + 2 \ln(\zeta_d). \end{aligned}$$

Since it depends neither on the hypothesis nor the bias, the term  $\gamma_r$  in (19) does not affect the optimal solution and can be ignored; thus, our optimization problem takes the final form

$$\min_{(H,d) \in \mathcal{H} \times \mathbb{R}^{2p}} c_r(d) + \sum_{(i,j) \in S} c_{ij}(d). \quad (20)$$

## REFERENCES

1. S. M. Herman and A. B. Poore, "Nonlinear least-squares estimation for sensor and navigation biases," in *Signal and Data Processing of Small Targets*, O. E. Drummond, ed., *SPIE* **6236**, 2006.
2. R. Y. Novoselov, S. M. Herman, S. M. Gadeleta, and A. B. Poore, "Mitigating the Effects of Residual Biases with Schmidt–Kalman Filtering," in *Eighth International Conference on Information Fusion*, (Philadelphia, PA), July 2005.
3. S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House, Norwood, MA, 1999.
4. M. Levedahl, "An explicit pattern matching assignment algorithm," in *Signal and Data Processing of Small Targets*, O. E. Drummond, ed., *SPIE* **4728**, 2002.
5. M. Levedahl, "Method and system for assigning observations." United States Patent US 7,092,924 B1, August 2006.
6. B. Borchers and J. E. Mitchell, "An improved branch and bound algorithm for mixed integer nonlinear programs," *Computers and Operations Research* **21**(4), pp. 359–367, 1994.
7. C. A. Floudas, *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*, Oxford University Press, New York, 1995.
8. I. E. Grossman, "Review of nonlinear mixed-integer and disjunctive programming techniques," *Optimization and Engineering* **3**, pp. 227–252, 2002.
9. M. Tawarmalani and N. V. Sahinidis, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming*, vol. 65 of *Nonconvex Optimization and Its Applications*, Kluwer Academic Publishers, 2002.

10. B. Borchers and J. E. Mitchell, "A computational comparison of branch and bound and outer approximation algorithms for 0-1 mixed integer nonlinear programs," *Computers and Operations Research* **24**(8), pp. 699–701, 1997.
11. R. Fletcher and S. Leyffer, "Numerical experience with lower bounds for miqp branch-and-bound," *SIAM J Optimization* **8**(2), pp. 604–616, 1998.
12. W. Zhang, "Depth-first branch-and-bound versus local search: A case study," in *Proc. 17-th National Conf. on Artificial Intelligence (AAAI-2000)*, pp. 930–935, AAAI, (Austin, TX), July 2000.
13. B. Kragel, S. Danford, S. Herman, and A. Poore, "Map probabilities for data association and absolute bias estimation with an arbitrary number of sensors," Technical Report TR-07-001, Numerica Corporation, August 2007.
14. S. Dandford, B. D. Kragel, and A. B. Poore, "Joint map bias estimation and association: Simulations," in *Signal and Data Processing of Small Targets*, O. E. Drummond, ed., *SPIE* **6699**, 2007.
15. S. Danford, B. Kragel, and A. Poore, "Joint map bias estimation and data association," Technical Report TR-07-002, Numerica Corporation, August 2007.
16. C. S. Adjiman, I. P. Androulakis, and C. A. Floudas, "Global optimization of mixed-integer nonlinear problems," *AIChE Journal* **46**(9), pp. 1769–1797, 2000.
17. K. Murty, "An algorithm for ranking all the assignments in order of increasing cost," *Operations Research* **16**, pp. 682–687, 1968.
18. S. Russell and P. Norvig, *Artificial Intelligence A Modern Approach*, Prentice Hall, Upper Saddle River, New Jersey 07458, second ed., 2003.
19. R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of  $a^*$ ," *Journal of the Association for Computing Machinery* **32**, pp. 505–536, July 1985.
20. W. Zhang and R. E. Korf, "Performance of linear-space search algorithms," *Artificial Intelligence* (79), pp. 241–292, 1995.
21. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Westley Publishing Company, Reading, Mass., 1974.
22. A. B. Poore, "Multidimensional assignment formulation of data association problems arising from multi-target tracking and multisensor data fusion," *Computational Optimization and Applications* **3**, pp. 27–57, 1994.