Simultaneous Track-to-Track Association and Bias Removal Using Multistart Local Search

Dimitri J. Papageorgiou and John-David Sergi
The Raytheon Company
Integrated Defense Systems
235 Presidential Way
Woburn, MA 01801-1060
{Dimitri_J_Papageorgiou, John-David_Sergi}@raytheon.com

Abstract—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. In this paper, we address the problem of simultaneously performing track-to-track association and bias estimation. We describe a polynomial-time multistart local search heuristic for quickly generating a number of high quality bias-assignment hypotheses. Computational results are presented to illustrate our findings and to compare our proposed algorithm to its competitors. Numerous test cases suggest that our multistart heuristic is suitable for real-time application and offers a practical and important "middle ground" for this problem by balancing performance and computation time.

TABLE OF CONTENTS

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. 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. Before information can be fused, the sensors must first ensure that the data 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, which is a focal point of this work, concerns error due to sensor biases. Biases resulting 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.

In this paper, we address the problem of simultaneously performing track-to-track association and bias estimation. This problem is known under several names in the literature including bias removal, data alignment, gridlock, sensor (data) registration, and global nearest pattern matching. For consistency, we will refer to the general problem as the sensor registration problem and a specific formulation, to be introduced in the next section, as the global nearest pattern matching (GNPM) problem, coined by Levedahl [9].

As discussed in [2] and [11], there are several potential sources of error that data alignment attempts to rectify. Moore and Blair [11] 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.

Finding correspondences between two sets of points such that the two sets are aligned with each other falls in a larger class of problems known as *point matching* in the pattern recognition, image alignment, and computer vision literature. Point matching problems are challenging computationally and arise in a diverse number of applications including sonar processing, finger print analysis, and localization of DNA markers [4].

The prototypical point matching problem involves two sets of observations – $N_A \in \Re^m$ and $N_B \in \Re^n$ – which must be aligned so that pertinent information can be extracted/inferred from the combined result. In order to determine an optimal correspondence, algorithms are typically interested in determining an affine mapping, i.e., identifying a linear transformation (rotation, scaling, or shear) and translation or shift term, that maps points in N_B onto those of N_A while minimizing some pre-defined distance metric. We emphasize this point because the sensor registration that we consider here deviates from this

1————————————
[1] 1-4244-1488-1/08/$25.00 ©2008 IEEE.
[2] IEEEAC paper #1142, Version 1, Revision 2, Updated Dec 11, 2007

goal in that a linear transformation is already known, while an optimal translation is sought. This is due to the fact that the locations of the sensors are typically known, biases are assumed constant or slowly varying with time, and therefore a straightforward transformation can be used to put the observations into the same coordinate frame. On the other hand, biases may be present in both range and angle dimensions for both sensors, so sensor registration attempts to estimate these biases and remove them before an association is made.

Figure 1 depicts a simple example of the point matching problem that sensor registration attempts to solve. 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 translational bias is such that no two corresponding track volumes overlap in the graphic.

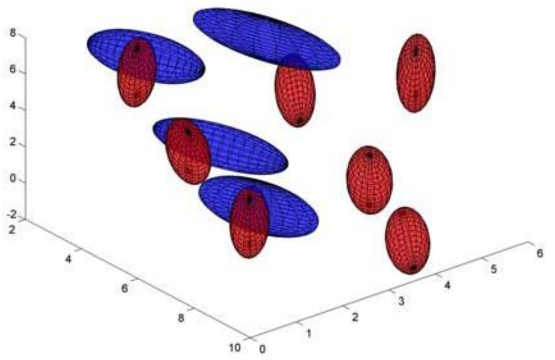

Figure 1 – The point matching problem in sensor registration using positional densities only.

In this paper, we focus solely on the problem of matching kinematic data from two sensor systems. This is not to say that one cannot use additional information to associate tracks. Indeed, there is ongoing effort to incorporate other relevant information (see, e.g., [16]) as it is recognized that additional feature information can significantly improve data association given missing observations, false alarms, and closely spaced objects. Feature estimation is also subject to bias issues, which sensor registration can handle. Although we never consider the feature-related problem specifically, we can easily incorporate it into our approach.

Contributions of this Paper

Although the sensor registration problem formulation has received adequate attention from a statistical estimation perspective [1], [2], [9], [16], relatively little has been described about the underlying optimization problem that must be solved. Our first contribution is to outline some of the theoretical underpinnings of this optimization problem and, in so doing, we reveal that a wealth of optimization methods are available to solve it.

As a second contribution, we introduce a polynomial-time multistart local search heuristic for generating a user-defined number of high quality bias-assignment hypotheses. The heuristic is a three-phased procedure that works well in practice. The first phase generates a set of "good" initial bias estimates, which seed the local search process. The second phase is a local search phase based on the "iterative bias estimation" method proposed by Blackman and Banh [1]. The third phase involves one or more applications of Murty's ranked assignment algorithm [12] for finding the *K* best assignments to a linear assignment problem.

The outline of the paper is as follows. In section 2, we formally introduce the GNPM problem, state several properties that will be useful in developing algorithms to solve it, and describe its mathematical structure from an optimization point of view. In section 3, we discuss algorithms to solve the GNPM problem optimally or suboptimally, and discuss our proposed algorithm. In section 4, we present computational results to illustrate our findings and to compare our proposed algorithm to its competitors. Concluding remarks and open questions for future research are discussed in section 5.

2. PROBLEM DESCRIPTION AND NOTATION

Let N_A and N_B denote the set of tracks (observations) made by sensor system *A* and *B*, respectively. Without loss of generality,we assume throughout that $n_A = |N_A|$, $n_B = |N_B|$, and $n_A \le n_B$. Tracks made by one sensor may not be present to the other sensor for several reasons, including geometry, a difference in sensor resolution, the presence of spurious observations (false alarms), and missed detections. Let $\mathbf{x}_i^A$ and $\mathbf{P}_i$, for $i \in N_A$, denote the state estimate and error covariance matrix, respectively, of the *i*th observation made by sensor *A*. Similarly, let $\mathbf{x}_j^B$ and $\mathbf{Q}_j$, for $j \in N_B$, denote the state estimate and covariance matrix, respectively, of the *j*th observation made by sensor *B*. 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.[3]

We denote a complete assignment by the vector $\mathbf{j}$. Consequently, the assignment of the *i*th observation in N_A with the *j*th observation in N_B is denoted by (i, j_i). It is possible that the *i*th observation in N_A is not assigned to any observation in N_B, in which case we still write (i, j_i), but

2

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

$j_i = 0$. We refer to such an assignment as a *null* assignment, or by saying that track i was assigned to the *dummy* track. A *nontrivial* assignment refers to an assignment with at least one non-null assignment. We denote each sensor's bias vector by $\mathbf{b}_A$ and $\mathbf{b}_B$ and the difference between the two biases is $\mathbf{b} = \mathbf{b}_A - \mathbf{b}_B$. Our prior knowledge of the bias emerges through our presumed/estimated registration covariance matrix $\mathbf{R} = E[\mathbf{b}\mathbf{b}^T]$. We refer to the pair $(\mathbf{b}, \mathbf{j})$ as a bias-assignment hypothesis, a hypothesis, or a solution to the GNPM problem.

Our objective is to simultaneously find the most likely track-to-track assignment and bias estimate. In order to do so, a likelihood function or cost function is needed to compare different solutions. As derived in [2] and [9], the likelihood function for the GNP matching problem is based upon the multivariate Gaussian density function. 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 $\mathbf{R}$) makes use of our prior knowledge/belief of sensor bias error to characterize the likelihood of a particular bias estimate. Namely, we assume that the expected difference between sensor A's bias and sensor B's bias is zero, which implies that $\mathbf{b}$ is simply a Gaussian vector with zero mean and covariance $\mathbf{R}$.[4] The second term consists of the sum 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 (sometimes called the generalized or normalized statistical 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 $\beta_{NTA} = \beta_N P_{\bar{A}B} + \beta_{FA}$ represents a target density of no target existing for sensor A, and $P_{\bar{A}B}$ is the probability of tracking an object with sensor B but not with sensor A; $\beta_{NTB} = \beta_T P_{A\bar{B}} + \beta_{FB}$ represents a target density of no target existing for sensor B, and $P_{A\bar{B}}$ 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.

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

$L(\mathbf{b}, \mathbf{j}) =$

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

For our purposes, it is more convenient from both a theoretical and computational vantage point to work with the negative log likelihood, which is valid since maximizing $L(\mathbf{b}, \mathbf{j})$ and minimizing $-\log(L(\mathbf{b}, \mathbf{j}))$ are equivalent, i.e., $(\mathbf{b}^*, \mathbf{j}^*)$ is a maximum of $L(\mathbf{b}, \mathbf{j})$ if and only if it is minimum of $-\log(L(\mathbf{b}, \mathbf{j}))$. In addition, since constants have no bearing on an optimal $(\mathbf{b}, \mathbf{j})$ pair, we remove the normalization term $\log\left((2\pi)^{D/2}\sqrt{|\mathbf{R}|}\right)$, multiply through by 2, and group remaining constants to obtain a modified version of the negative log likelihood function:

$$NLL(\mathbf{b}, \mathbf{j}) = \mathbf{b}^T\mathbf{R}^{-1}\mathbf{b} + \sum_{i=1}^{n_A} \left\{ \begin{array}{ll} d_{ij_i}^2(\mathbf{b}) + \log\left(|\mathbf{S}_{ij_i}|\right) & \text{if } j_i > 0 \\ g & \text{if } j_i = 0 \end{array} \right\} \quad (2)$$

where

$$g = \log\left(\frac{B_T P_{AB}}{B_{NTA} B_{NTB} (2\pi)^D}\right)$$

is a so-called (log likelihood) *gate value*, a threshold used to compare the likelihood of assigning a track $i \in N_A$ to a track $j \in N_B$ in favor of making a null assignment, i.e., assigning track i to the dummy track $j = 0$. A large gate value reflects a strong belief that each track in N_A corresponds to a track in N_B. On the other hand, a small gate value suggests that if, after accounting for the bias, no unassigned track $j \in N_B$ is close enough to track $i \in N_A$, i.e., $d_{ij}^2(\mathbf{b}) + \log\left(|\mathbf{S}_{ij}|\right) > g$, then a null assignment is preferred. The determination of an optimal gate value remains an open question.

3

[4] 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, 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 radars are different.

Until this point, we have left out an important detail regarding track-to-track assignments. The standard assumption in track-to-track association, which we follow, is that each track $i \in N_A$ can be assigned to at most one track $j \in N_B$ and vice versa. Thus, at the heart of the GNPM problem is a constrained optimization problem in which we are interested in maximizing the likelihood function $L(\mathbf{b}, \mathbf{j})$ [equivalently, minimizing $NLL(\mathbf{b}, \mathbf{j})$] subject to the constraints that each observation in N_A is assigned to at most one observation in N_B and each observation in N_B is assigned to at most observation in N_A.

To make this formulation rigorous from a mathematical programming perspective, we introduce 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 track $i \in N_A$ is assigned to track $j \in N_B \cup \{0\}$, and is zero otherwise. As was recognized by Hirsch [6], we can now cast the GNPM problem as a nonlinear mixed-integer optimization problem (NMIP for short):

$$z_{NLL} =$$

$$\min_{\mathbf{b},\mathbf{y}} \ \mathbf{b}^T \mathbf{R}^{-1} \mathbf{b} + \sum_{i=1}^{n_A} \sum_{j=0}^{n_B} \begin{Bmatrix} d_{ij}^2(\mathbf{b}) + \log\left(\left|\mathbf{S}_{ij}\right|\right) & \text{if } j > 0 \\ g & \text{if } j = 0 \end{Bmatrix} y_{ij}$$

$$\begin{aligned} \text{s.t.} \ & \sum_{j=0}^{n_B} y_{ij} = 1, \text{for } i = 1, \ldots, n_A \\ & \sum_{i=1}^{n_A} y_{ij} \le 1, \text{for } j = 1, \ldots, n_B \qquad (3) \\ & y_{ij} \in \{0,1\}, \text{for } i = 1, \ldots, n_A, \text{for } j = 0, \ldots, n_B \\ & \mathbf{b} \in \Re^D. \end{aligned}$$

The problem is nonlinear because the objective function is nonconvex, as shown in Figure 2, due to the cubic terms of the form $\mathbf{b}^T\mathbf{b}y_{ij}$ that arise in the summation. In addition, because our decision variables for the bias components are real numbers (hence, continuous), while the assignment decision variables y_{ij} are binary, the problem is called mixed-integer, i.e., a mixture of both continuous and discrete decision variables. The constraints on the decision variables y_{ij} are the same as those found in the (asymmetric) linear assignment problem.

After defining the set $\mathrm{Y} = \{\mathbf{y} \in \{0,1\}^{n_A \times (n_B+1)} \mid \sum_{j=0}^{n_B} y_{ij} = 1,$ for $i = 1, \ldots, n_A, \sum_{i=1}^{n_A} y_{ij} \le 1$, for $i = 1, \ldots, n_B\}$ of all feasible assignments, we can re-write problem (3) in the following compact form:

$$z_{NLL} = \min_{\mathbf{b} \in \Re^D, \mathbf{y} \in \mathrm{Y}} \{f(\mathbf{b}) + g(\mathbf{b}, \mathbf{y})\} \qquad (4)$$

where $f(\mathbf{b}) = \mathbf{b}^T \mathbf{R}^{-1} \mathbf{b}$, $g(\mathbf{b}, \mathbf{y}) = \sum_{i=1}^{n_A} \sum_{j=0}^{n_B} c_{ij}(\mathbf{b}) y_{ij}$, $\mathbf{y}$ is a vector of the y_{ij} decision variables, and

$$c_{ij}(\mathbf{b}) = \begin{Bmatrix} d_{ij}^2(\mathbf{b}) + \log\left(\left|\mathbf{S}_{ij}\right|\right) & \text{if } j > 0 \\ g & \text{if } j = 0 \end{Bmatrix}.$$

We will use this compact formulation in the next section to point out some interesting properties that various solution methods can exploit.

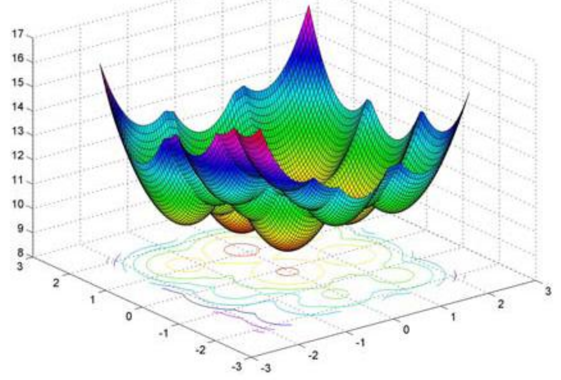

Figure 2 – An example of a highly nonconvex GNPM likelihood function for a 2D problem instance involving 4 sensor *A* tracks and 16 sensor *B* tracks. The 2D bias is shown in the horizontal plane, while the minimum negative log likelihood is shown along the vertical axis.

As an aside, we favor the NMIP formulation over the following combinatorial-based formulation:

$$\min_{\mathbf{b} \in \Re^D, \mathbf{j} \in \Pi(n_A, n_B)} \mathbf{b}^T \mathbf{R}^{-1} \mathbf{b} + \sum_{i=1}^{n_A} \begin{Bmatrix} d_{ij_i}^2(\mathbf{b}) + \log\left(\left|\mathbf{S}_{ij_i}\right|\right) & \text{if } j_i > 0 \\ g & \text{if } j_i = 0 \end{Bmatrix},$$

where $\Pi(n_A, n_B)$ is the set of all feasible assignment vectors, primarily to emphasize that our local search algorithm is not based on combinatorial arguments, but also because the NMIP formulation is more common in the mathematical programming community.

Observations

We now describe four observations about the GNPM problem. The first two properties will be useful once we begin discussing algorithms as they can be applied in an iterative manner that forms the basis of our local search procedure. The last two observations highlight characteristics of the solution space and the number of optimal solutions.

Observation 1 [Given a bias **b***, we can efficiently compute an optimal assignment* $\mathbf{j}^*(\mathbf{b})$*]*—Given a bias estimate **b**, the track-to-track association problem within the GNPM problem as stated in equation (4) is

$$g(\mathbf{b}) = \min_{\mathbf{y}\in \mathrm{Y}} g(\mathbf{b}, \mathbf{y})$$

and can be solved in strongly polynomial time to obtain an optimal assignment vector, which we denote by $\mathbf{j}^*(\mathbf{b})$, to explicitly show its dependence on the given bias estimate **b**. In particular, $g(\mathbf{b})$ is a two-dimensional (asymmetric) linear assignment problem having the following form when cast as a linear optimization problem:

$$\begin{aligned} g(\mathbf{b}) = \min_{\mathbf{y}} \;& \sum_{i=1}^{n_A}\sum_{j=0}^{n_B} c_{ij}(\mathbf{b})y_{ij} \\ \text{s.t.} \;& \sum_{j=0}^{n_B} y_{ij} = 1, \text{for } i = 1,\ldots,n_A \\ & \sum_{i=1}^{n_A} y_{ij} \le 1, \text{for } j = 1,\ldots,n_B \\ & 0 \le y_{ij} \le 1, \text{for } i = 1,\ldots,n_A, \text{for } j = 0,\ldots,n_B. \end{aligned}$$

This problem is solvable in $O(n_A^2 n_B)$ time using the Jonkers-Volgenant-Castañon (JVC) algorithm [5], [8] or the successive shortest path algorithm [15]. This observation is not profound and should make intuitive sense since if the bias is known, the GNPM problem reduces to a standard track-to-track association problem, which is customarily modeled as a linear assignment problem (see, e.g., [2] chapter 9.6).

Observation 2 [Given an assignment **j***, we can efficiently compute an optimal bias* $\mathbf{b}^*(\mathbf{j})$*]*—Given a nontrivial assignment vector **j**, we can easily determine the optimal bias estimate $\mathbf{b}^*(\mathbf{j})$ for that assignment by taking the partial derivative of equation (2) with respect to **b** and setting the result equal to zero. The resulting expression is a system of D linear equations in D unknowns

$$\mathbf{Ab} = \sum_{i=1}^{n_A} \begin{Bmatrix} \mathbf{S}_{ij_i}^{-1}\left(\mathbf{x}_i^A - \mathbf{x}_{j_i}^B\right) & \text{if } j_i > 0 \\ \mathbf{0} & \text{if } j_i = 0 \end{Bmatrix} \tag{5}$$

where

$$\mathbf{A} = \mathbf{R}^{-1} + \sum_{i=1}^{n_A} \begin{Bmatrix} \mathbf{S}_{ij_i}^{-1} & \text{if } j_i > 0 \\ \mathbf{0} & \text{if } j_i = 0 \end{Bmatrix}.$$

Notice that only assigned observations contribute to the bias estimate and that $\mathbf{b}^*(\mathbf{j})$ is unique since the matrix **A** is symmetric positive definite and, therefore, invertible. With no assignments made, the bias is indeterminate and $\mathbf{b} = \mathbf{0}$ is the maximum likelihood estimate.

Observation 3 [The solution space of all assignments grows factorially in the number of tracks to associate]—Our goal is to assign the tracks in N_A with those in N_B or possibly with the dummy track. Once all tracks in N_A are assigned, we terminate. There are $\binom{n_A}{k}$ ways of choosing k tracks from N_A that will be assigned to a track in N_B. For each set of k tracks, there are n_B ways to assign the first track, $n_B - 1$ ways to assign the second, and so on. Hence, there are $n_B!/(n_B - k)!$ ways of assigning the set of k tracks. In total, the number of possible assignments is

$$|\Pi(n_A, n_B)| = \sum_{k=0}^{n_A} \binom{n_A}{k} \frac{n_B!}{(n_B - k)!}. \tag{6}$$

Note that this is the exact same number of possible assignments to an asymmetric linear assignment problem in which null assignments are permitted.

Observation 4 [Multiple global optima may exist]—The GNPM problem can have multiple global optima. For a simple example, consider a problem in one dimension in which set N_A has two observations with positions x_1^A = -1 and x_2^A = 1. Meanwhile, set N_B has three observations with positions x_1^B = -2, x_2^B = 0, and x_3^B = 2. Suppose the variance in each observation is one-half and the variance in the bias estimate is one. Then, there are two globally optimal bias-assignment hypotheses: (2/3, {(1,1),(2,2)}) and (-2/3, {(1,2),(2,3)}). Although multiple global minima are rare, this example illustrates an important point: there may exist multiple solutions with a high likelihood that should be monitored before making an irrevocable track-to-track association.

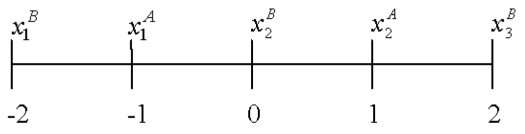

Figure 3 – A 1D example illustrating that multiple global optima may exist.

3. Solution Methods

In this section, we discuss solution methods for the GNPM problem. We distinguish between two broad classes of algorithms: exact methods and heuristics. An exact method is guaranteed to converge to a globally optimal solution to an optimization problem, provided "sufficient" (possibly infinite) time and memory are available. On the other hand, heuristic methods do not provide an optimality guarantee, but are nevertheless crucial for real-time systems. Indeed, as tracking systems evolve and are required to process data for an increasing number of tracks all within a matter of seconds, obtaining high quality solutions quickly is arguably more important than achieving optimality.

We restrict our attention to methods for solving the GNPM problem and not sensor registration algorithms, in general. Indeed, other techniques, such as covariance inflation, clustering closely spaced objects, and ignoring bias altogether have been tried. The survey by Brown [3] covers a vast number of algorithms that have been applied to more general image alignment problems.

Our primary objective in solving the GNPM problem is to obtain a globally optimal solution, i.e., an optimal bias-assignment hypothesis $(\mathbf{b}^*, \mathbf{j}^*)$. At the same time, since knowledge of the true translational bias or a global optimum to the GNPM problem does not guarantee that the correct track-to-track association will be made, we are often faced with a secondary goal of finding the K best solutions to the GNPM problem so that future decisions can account for possible misassociations. In particular, if there is a unique global optimum whose likelihood can be deemed "sufficiently better" than the next best solution, then we typically have more confidence in making an irrevocable decision that this bias-assignment hypothesis is the correct one. On the other hand, if there are multiple high quality solutions whose likelihoods differ by negligible amounts, as can often happen for a large number of closely-spaced tracks or when differences in sensor resolution cause one sensor to have more tracks than the other, we typically have less confidence in the global optimum. By comparing the K best solutions, we can determine whether or not there exists a subset of track-to-track associations that remain the same in all K solutions. If so, we may wish to report a high confidence in the associations made for this subset of tracks and a low confidence for the remaining tracks that are competing to be associated with one another. It then may be possible for multiple hypotheses to be maintained by the tracking system in hopes that future information will resolve these ambiguities. In what follows, we discuss solution methods that address both of these objectives.

3.1 Exact Methods

Perhaps the most obvious exact method is an exhaustive enumeration of all possible assignments. Since there is an optimal bias estimate associated with each assignment, as shown in equation (5), an algorithm that enumerates all assignments will produce a globally optimal solution to the GNPM problem. However, as shown in equation (6), this number grows factorially in the number of tracks to associate and, therefore, renders exhaustive enumeration prohibitively expensive even for problem instances of moderate size. Nevertheless, it is worth formally outlining how such an algorithm would behave to contrast it with other methods.

A systematic way of generating all possible assignments is to a create search tree or enumeration tree. The tree consists of nodes at varying depths, where each node represents a partial assignment or complete assignment. Without loss of generality, we assume that tracks in N_A are to be assigned in increasing order to tracks in N_B, 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 assignments 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 assignment of track 1 in N_A with all n_B tracks in N_B plus the null partial assignment $\{(1,0)\}$. From each node at depth 1 with an assignment of the form $\{(1, j_1)\}$ with j_1 in N_B, n_B branches are created. Each of these nodes has a partial assignment $\{(1, j_1), (2, j_2)\}$, where $j_1 = 1, \ldots, n_B$ and $j_2 \in \{0, \ldots, n_B\} \setminus \{j_1\}$. From the node with partial assignment $\{(1,0)\}$, $n_B + 1$ branches are created since track 2 can be assigned with any track in N_B plus the dummy track. This branching continues until a depth n_A is reached, at which point there exist $|\Pi(n_A, n_B)|$ *leaf* nodes, as shown in equation (6). For each leaf node (except the one with the trivial assignment), we can compute the best bias estimate $\mathbf{b}^*(\mathbf{j})$ for that node's assignment $\mathbf{j}$, and hence calculate the likelihood of that bias-assignment pair.

An alternative approach to exhaustive enumeration is known under several names, including implicit enumeration, *branch-and-bound* (the term we will use), strategic partitioning, and provisional cutting, depending on the implementation. Branch-and-bound algorithms pervade the field of integer programming and combinatorial optimization and are discussed in virtually every introductory textbook on the subject (see, e.g., [13]). Branch-and-bound is founded upon the idea of exploring nodes (or partial assignments) in the search tree in an intelligent manner so that not all combinations need be examined. In the context of the GNPM problem, the term *branch* refers to the manner in which partial assignments are constructed, i.e., the partitioning of the solution space. The term *bound* refers to deterministic bounds that are computed during the search process, which can be used to prune (sometimes called "fathom") partial assignments 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. Pruning nodes is absolutely essential when solving large problem instances using branch-and-bound because it reduces the time and memory needed to explore the search tree. Tight bounds are crucial in facilitating this pruning process. Two such methods for pruning are now described.

Let $\mathbf{p}$ be a partial assignment of a node at depth k, for $k = 1, \ldots, n_A$, i.e., $\mathbf{p}$ corresponds to an assignment

$\{(1,p_1),\dots,(k,p_k)\}$. Let $h(\mathbf{b},\mathbf{p})$ denote the GNPM negative log likelihood function without the bias term $\mathbf{b}^T\mathbf{R}^{-1}\mathbf{b}$. That is,

$$h(\mathbf{b},\mathbf{p}) = \sum_{i=1}^{k} \begin{cases} d_{ip_i}^2(\mathbf{b}) + \log\left(\left|\mathbf{S}_{ip_i}\right|\right) & \text{if } p_i > 0 \\ g & \text{if } p_i = 0 \end{cases}$$

Finally, let $\hat{\mathbf{b}}(\mathbf{p})$ be the unique bias estimate that minimizes $h(\mathbf{b},\mathbf{p})$ for a given partial assignment **p**, which can be found by solving a system of linear equations nearly identical to equation (5). One simple test for pruning a node with partial assignment **p** is to make use of the gate value *g*. In particular, if $h(\hat{\mathbf{b}}(\mathbf{p}),\mathbf{p}) > kg$, then we can prune this node because tracks $i = 1,\dots,k \in N_A$ could be assigned to the dummy track at lower cost (a cost of *kg*) for this and for all other bias estimates. Note that this pruning test is highly dependent on the gate value *g*; the larger the gate value, the less frequently this test will lead to a node being pruned, and, as a consequence, more nodes will be kept in memory.

When solving for a single global optimum, a node may be pruned the moment its lower bound exceeds the best upper bound on the GNPM negative log likelihood function. A valid upper bound is the minimum negative log likelihood over all bias-assignment pairs $(\mathbf{b}^*(\mathbf{j}),\mathbf{j})$ found thus far in the search, where **j** is a complete feasible assignment with corresponding bias $\mathbf{b}^*(\mathbf{j})$ as determined by equation (5). A valid lower bound for a node at depth *k*, for $k = 1,\dots,n_A$, (see the appendix for a proof) is

$$h(\hat{\mathbf{b}}(\mathbf{p}),\mathbf{p}) + \min_{\mathbf{y}\in\Upsilon(k)} \sum_{i=k+1}^{n_A}\sum_{j=0}^{n_B} \begin{cases} \log\left(\left|\mathbf{S}_{ij}\right|\right) & \text{if } j > 0 \\ g & \text{if } j = 0 \end{cases} y_{ij}$$

where $\Upsilon(k)$ denotes the set of all feasible matchings of unassigned tracks $i \in N_A$ $(i = k+1,\dots,n_A)$ and all unassigned tracks $j \in N_B$ remaining at depth *k*. With this definition of $\Upsilon(k)$, the right-most term is just a linear assignment problem over the unassigned tracks in N_A and N_B. Since solving a linear assignment problem for every node may be costly from a computational standpoint, one could replace the right most term with a more conservative bound, such as

$$\sum_{i=k+1}^{n_A} \min\left\{g, \min\left\{\log\left(\left|\mathbf{S}_{ij}\right|\right) \mid j = 1,\dots,n_B; j \text{ unassigned}\right\}\right\}.$$

Unfortunately, when solving for the *K* best solutions, this pruning test cannot be used because one of the *K* best solutions might be inadvertently pruned.

3.2 Heuristics

Because there are an infinite number of heuristics that could be applied to the GNPM problem to obtain a suboptimal solution, we now briefly describe some of the more promising approaches that have been implemented or that could be implemented in the future.

Keeping within the tree search framework, one possible extension when memory becomes scarce is to begin pruning nodes that have a low probability of containing a global optimal solution. Since this pruning takes place prior to knowing with absolute certainty that a global optimum resides in a leaf node of that parent node, the algorithm is no longer guaranteed to converge to an optimal solution. Nevertheless, if premature pruning is done intelligently, one can maintain a high probability of obtaining a global minimum. The drawback of this approach is that the algorithm is still not polynomial in the size of the problem instance. This approach can be used for virtually any branch-and-bound algorithm.

A fundamentally different approach for attacking the GNPM problem is to use so-called *decomposition methods*. Decomposition methods are widespread in the field of optimization as they permit one to exploit the structure of a problem by decomposing it into smaller, more manageable parts that typically exhibit a structure that is more amenable to efficient solution procedures. They are also a powerful tool for solving and/or approximating large-scale problems in a reasonable amount of time.

As was noted in section 2 equation (4), the GNPM problem can be formulated as a nonlinear mixed-integer optimization problem in the following compact form:

$$z_{NLL} = \min_{\mathbf{b}\in\Re^D,\mathbf{y}\in\Upsilon} \{f(\mathbf{b}) + g(\mathbf{b},\mathbf{y})\}. \qquad (7)$$

Since simultaneously optimizing over both $\mathbf{b}\in\Re^D$ and $\mathbf{y}\in\Upsilon$ is challenging, it is sometimes beneficial to decompose the optimization problem into two problems as follows:

$$z_{NLL} = \min_{\mathbf{b}\in\Re^D}\left\{f(\mathbf{b}) + \min_{\mathbf{y}\in\Upsilon} g(\mathbf{b},\mathbf{y})\right\}. \qquad (8)$$

The difference between problems (7) and (8) is subtle, but important. Whereas in (7) the minimization takes place simultaneously over the bias **b** and the assignment decision variables y_{ij}, in (8) we have an outer and an inner (or nested) minimization problem. That is, we first minimize over the bias vector $\mathbf{b}\in\Re^D$ and subsequently minimize over the function $g(\mathbf{b})$, where **b** is now fixed.

Rather than minimizing over all $\mathbf{b}\in\Re^D$ in problem (8), we now describe a local search method, due to Blackman and

Banh [1], that begins with an initial bias estimate $\mathbf{b}_0$ and applies the results of observations 1 and 2 from section 2 in an iterative manner. An appealing way of understanding this local search procedure is to imagine a two-player cooperative game in which both players wish to find a local minimum of the function $f(\mathbf{b}) + \min\{g(\mathbf{b},\mathbf{y}) \mid \mathbf{y} \in \mathrm{Y}\}$. Player 1 controls the choice of the bias estimate, while player 2 controls the choice of the assignment. Player 1 plays first by selecting a bias $\mathbf{b}_0 \in \Re^D$. Player 2, armed with full knowledge of the bias estimate chosen, solves the inner minimization problem $g(\mathbf{b}_0)$, which is nothing but a linear assignment problem as shown in observation 1, for an assignment vector $\mathbf{j}^*(\mathbf{b}_0)$. (Note the dependence of the optimal assignment on the original bias estimate $\mathbf{b}_0$). When play returns to player 1, she uses observation 2 to compute the unique bias estimate $\mathbf{b}^*(\mathbf{j}^*(\mathbf{b}_0))$ that minimizes the cost of that assignment by solving the system in equation (5). Play continues until player 1 encounters the same bias estimate as she computed in the previous round. We refer to this iterative procedure between computing optimal assignments and optimal biases as `LocalSearch`, since it terminates once a local minimum to the GNPM problem is found.

3.3 A Multistart Local Search Heuristic

The difficulty with the GNPM problem is the nonconvexity of the GNPM (negative log) likelihood function, which becomes rife with local minima as the number of tracks increases. Meanwhile, the proximity of the local minima increases as the target density grows. Motivated by the growing presence of local minima and the availability of a fast local search method, we propose a so-called *multistart* local search algorithm, in which a list *biasList* = $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$ of initial bias estimates are generated and then used to seed `LocalSearch`. Multistart techniques are commonplace in nonconvex nonlinear optimization, but have been conspicuously overlooked for the GNPM problem.

Fundamental questions regarding multistart methods include: (1) How many starting points (or *seeds*) should be generated? (2) How should seeds be selected? On the one hand, one could conceivably scour $\Re^D$ with initial bias estimates, but this would likely result in redundant explorations of basins that lead to the same local minima. On the other hand, if initial bias estimates are few and/or chosen too far apart, there is a risk of bypassing a basin in which a global minimum resides. Thus, this tradeoff must be balanced judiciously.

The number of starting points is primarily dictated by computation time requirements. For our purposes, we choose 30 starting points based on a self-imposed requirement of (roughly) one second of computation time and empirical results of worst-case run times for the local search algorithm. As the number of tracks increase, we believe that even 30 seeds will require too much time. Regarding the construction of starting points, there are numerous methods for generating a list of initial bias estimates. We describe the three methods that we have implemented and tested:

Gaussian(**0**,**R**)

Because the bias is assumed to be a multivariate Gaussian(**0**,**R**) random variable, a naive way of generating multiple initial bias estimates is to generate them randomly from a Gaussian(**0**,**R**) distribution. As the number of randomly generated seeds grows large, so too does the probability of finding a seed that will lead to a global minimum. Despite generating a small number of initial bias estimates, our results show that this approach is very fast and performs reasonably well for the scenarios we considered.

Fast Fourier Transform Correlation

Another approach for finding a list of "good" initial bias estimates is what Stone et al. [16] term *FFT correlation*, as it takes advantage of the computational efficiency of the Fast Fourier Transform (FFT) for performing correlation, and has been successfully applied in the computer vision literature [3]. The FFT correlation method that we employ works as follows: We assume that the position of each track is specified in a 3-dimensional Cartesian coordinate frame. We next create discrete approximations M_A and M_B for each sensor's target densities by constructing a 3-dimensional grid containing N^3 cells, where N is a user-defined parameter typically chosen as a power of 2. For each sensor, we count the number of tracks that fall in each cell. Thus, $M_A(\mathbf{x})$ denotes the number of sensor A tracks in cell $\mathbf{x}$. For each translational bias $\mathbf{t}$ in the discrete space, we compute the correlation function

$$\Gamma(\mathbf{t}) = \sum_{\mathbf{x}} M_A(\mathbf{x}-\mathbf{t}) M_B(\mathbf{x}) .$$

Finally, we take the highest "peaks" of the correlation function to be the initial bias estimates used to seed the local search algorithm. Another option, which we did not explore, is to use a coarser approximation by keeping N small and then choose multiple biases from the same peak of the correlation function. We compute an initial bias estimate from the position elements only. Velocity components could also be used, but the tradeoff is increased computation time. The complexity of the FFT (in a single dimension) is $O(N \log N)$. Consequently, the computation time increases as the number of dimensions and cells increase.

Reduced Branch-and-Bound

The final approach we consider for generating a list of seeds is to apply a limited (or reduced) branch-and-bound in which only a subset N'_A of tracks in N_A are assigned to tracks in N_B (all tracks in N_B are considered). This approach has two advantages. First, when tracks are well separated, branch-and-bound is typically able to prune many nodes at a shallow depth and leave only a small portion of the search tree unexplored. (The problem is, in general, we cannot say a priori how difficult the problem instance will be.) Second, the likelihood of the best solution found using the reduced branch-and-bound provides a lower bound on the likelihood of the best solution to the full GNPM problem. The difference between likelihoods of the best solution to the full problem and that of the reduced problem is known as the *duality gap*. This gap provides us with a degree of suboptimality for the solution found using local search. The drawback of this approach is that the quality of the initial bias estimates, in general, depends on the depth of the branch-and-bound considered as well as on the choice and ordering of the tracks in N'_A. As the depth increases, so too does computation time and memory. This approach becomes increasingly limited as the number of tracks increases. Levedahl [9] suggests using points on the convex hull defined by N_A. Finally, the theoretical downside of this approach is that, because even reduced branch-and-bound is not a polynomial-time algorithm, the complexity of the entire algorithm would not be polynomial.

3.4 Murty's Ranked Assignment Algorithm

The final phase of our heuristic is only necessary if one wishes to find K near optimal solutions to the GNPM problem and requires one or more applications of Murty's ranked assignment algorithm [12]. As previously stated, even if the true bias were known, there is still the possibility of making an incorrect association. This potential for misassociation motivates our interest in an algorithm that can find the K best assignments to use for improved decision making. Murty's algorithm accomplishes precisely this objective in a systematic manner and in polynomial time.

Murty's algorithm, described in detail in [12], [14], [15], finds the K best assignments to the classic two-dimensional linear assignment problem (LAP for short). For our purposes, we apply the method from the best bias estimate associated with each local minimum until the K "best" assignments (over all of the local minima that we identified) are computed. Specifically, we order the local minima found previously in the algorithm in nondecreasing order according to their GNP (negative log) likelihood. We create a list *solnList* to hold the K best solutions. We assume that this list is kept sorted in nondecreasing order according to likelihood. For each local minimum $(\mathbf{b}, \mathbf{j})$, we compute the K best assignments associated with the bias estimate $\mathbf{b}$ using Murty's algorithm. Once the likelihood of a local minimum exceeds the Kth best solution in *solnList*, we stop.

Since Murty's algorithm is well documented, we omit the finer points of the algorithm here and only mention that we implemented the three optimizations to Murty's algorithm discussed in Miller et al. [10]: (1) inheriting dual variables and partial solutions during partitioning; (2) sorting subproblems by lower cost bounds before re-solving; and (3) partitioning in an optimized order.

It is worth noting that, to the best of our knowledge, the re-optimization step, discussed in [10] and [15], in which a single shortest augmenting path problem from an unassigned track in N_A to an unassigned track in N_B is only guaranteed to generate a primal feasible solution for which complementary slackness conditions hold when solving symmetric LAPs. This has two consequences. First, after solving a single shortest augmenting path problem (as is done with symmetric LAPs), we check if strong duality holds. If not (which occurred in fewer than 10% of the instances we encountered), we re-solve the LAP from scratch. Second, this means that the worst case complexity of Murty's algorithm for the asymmetric LAPs that we encounter remains $O(Kn_B^4)$. Empirically, we have found that average run times are nearly identical to what one would expect in the symmetric case.

4. Performance Studies

4.1 Simulation Environment

A simulation testbed was created in MATLAB to compare performance of various algorithms. For simplicity, no tracking algorithms were used. Instead, the track sets N_A and N_B are randomly generated to represent a "snapshot in time" based on the following criteria:

(1) track set size for each sensor (n_A and n_B),

(2) track set overlap, i.e., the number n_C of tracks the two sensors have in common, and

(3) object density (how closely spaced the objects are with respect to their covariance errors).

Track states are maintained in a 6-dimensional Cartesian reference frame $(x, y, z, \dot{x}, \dot{y}, \dot{z})$. All covariance matrices are diagonal matrices. The $\mathbf{P}$ and $\mathbf{Q}$ matrices have a 1-σ position error fixed at 100 meters and 1-σ velocity error fixed at 1 m/s. As for the $\mathbf{R}$ matrix, the 1-σ error in positional bias is 400 meters. The 1-σ error in velocity bias is a very small positive constant and is therefore negligible. The errors in velocity bias are intentionally kept small so at to limit the contribution of the velocity dimension in the association process. These values are kept constant over all runs.

A particular problem instance is created as follows: After choosing n_A and n_B (with $n_A \leq n_B$), $n_T = |N_A \cup N_B|$ truth objects are created by randomly generating position components (uniformly) in a cube with a given side length. Velocity components are generated in a similar manner with a "velocity" cube. The side length of the two cubes determines how closely spaced the objects are to one another. Next, a user-defined number $n_C = |N_A \cap N_B|$ of common tracks are identified. The true bias is drawn randomly from a Gaussian($\mathbf{0}$,$\mathbf{R}$) distribution and half of the bias is added to each track in N_A, while the other half is subtracted from each track in N_B. Finally, each track in N_A (respectively, N_B) receives random measurement error, which is drawn randomly from a Gaussian($\mathbf{0}$,$\mathbf{P}$) [Gaussian($\mathbf{0}$,$\mathbf{Q}$)] distribution.

4.2 Algorithms Compared

All algorithms compared solve the GNPM problem for the $K = 30$ best solutions and all multistart local search variants use 30 initial bias estimates to seed the local search method. The algorithms compared are:

(1) [GNPL] Implemented by Levedahl [9], GNPL is an inexact branch-and-bound method, which has proven to be quite effective for solving small problem instances in a reasonable amount of time (i.e., a few seconds). It is "inexact" in the sense that it is not guaranteed to find an optimal solution. On the other hand, it behaves like the branch-and-bound algorithm described in section 3 except with a simpler and faster gating test, which avoids calculating a bias estimate $\hat{\mathbf{b}}(\mathbf{p})$ for every node. Note that because all algorithms are asked to solve for the K best solutions, GNPL only prunes nodes based on a gating test.

(2) [Gaussian($\mathbf{0}$,$\mathbf{R}$) + local search] Thirty seeds are randomly generated from a multivariate Gaussian($\mathbf{0}$,$\mathbf{R}$) distribution.

(3) [FFT Correlation + local search] We use the FFT correlation method described in section 3 to compute initial bias estimates in position only. Initial velocity bias estimates are all zero. A small value of $N = 16$, corresponding to a grid with 4096 cells, was used.

(4) [Reduced GNPL + local search] Only the first 5 tracks in N_A are assigned using the GNPL algorithm. The 30 best bias estimates generated from this reduced problem are then used to seed the local search heuristic.

(5) ["IJVC" or single start local search] This method performs a single local search with an initial bias $\mathbf{b} = \mathbf{0}$.

After finding one or more local minima, algorithms 2-5 finish by performing Murty's ranked assignment algorithm as described in section 3.4 in order to find the K "best" solutions. Due to memory limitations, an exact branch-and-bound algorithm as described in section 3 was out of the question (even when solving for the single best solution only). However, we mention in passing that the Gaussian($\mathbf{0}$,$\mathbf{R}$) variant of the heuristic found the global minimum in just over 90% of the simulations we conducted for instances involving 10 on 10 tracks with all 10 common tracks in common.

All tests were performed on a laptop computer with a 1.83GHz Intel T2400 processor running MS Windows XP Version 2002. The multistart local search heuristic, Murty's ranked assignment algorithm, and the linear assignment problem solver (the JVC algorithm) were coded in Java by the first author. The LAP code used a cost matrix implementation. Potential savings are available if a sparse network representation were used. We used the Java Matrix Package (JAMA) version 1.0.2, a basic linear algebra package for Java, available on the web [7], to perform matrix operations. GNPL was coded in C++ by Levedahl. The FFT correlation method was coded in MATLAB by the first author.

4.3 Performance Metrics

The measures of performance that we collected include:

(1) Probability of Correct Association (PCA) averaged over 200 Monte Carlo runs. The fraction of correct associations on a given run is determined by comparing the true track-to-track association with the association vector of the best solution produced by each of the algorithms, and computing the number of correct associations made divided by n_A.

(2) Average CPU time for algorithm execution (including problem instantiation).

(3) Maximum CPU time for algorithm execution.

A suitable metric for assessing target density is the average nearest neighbor positional distance normalized to the one-sigma random measurement error. In other words, we gauge scene density by looping over each track i in N_A, finding the track $j \neq i$ in N_A for which the Mahalanobis distance (in position only) between i and j is a minimum, and summing up these distances. After performing the same calculation for track set N_B, we take the average to produce a unitless number NN_p/σ_p that represents the overall track spacing or scene density. In general, as the ratio NN_p/σ_p decreases, targets become more closely spaced (with respect to the random measurement errors) and thus more difficult to reliably associate.

4.4 Computational Results

The following results are intended to compare the algorithms based on the performance metrics described above. In general, the probability of correct association is low because the problem instances generated are intentionally challenging. Overall performance could improve by using an adaptive gate setting and/or feature information. Since our primary goal is to show that our multistart heuristic is competitive with other algorithms for any parameters settings, the log likelihood gate value g was fixed at 10 throughout. This value was chosen as it works well when all tracks seen by sensor A are also seen by sensor B. On the other hand, it does not work well when the number of common tracks is small.

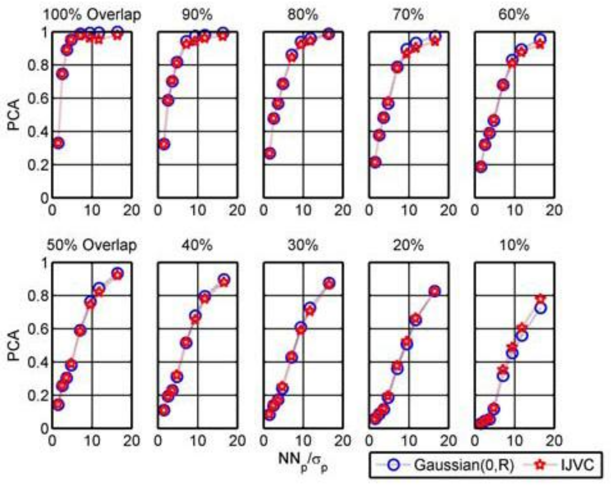

Figure 4 – Average probability of correct association (PCA) improves as tracks separate

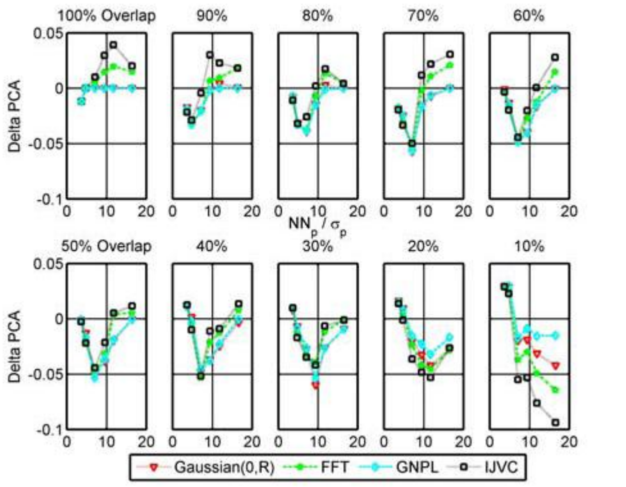

Figure 5 – Delta PCA between GNPL and the other four algorithms

Figures 4-6 apply to a 20 on 20 scenario, i.e., $n_A = n_B = 20$. Figure 4 shows the average probability of correct association (PCA) for varying levels of track set overlap (100% means both sensors see the same 20 objects, 90% means 18 of the 20 objects are seen by both sensors, and so on) and decreasing scene density. Consistent with our intuition, the probability of correct association is relatively poor when objects are closely spaced, but increases rapidly as they separate. Furthermore, PCA degrades as the number of tracks in common decreases. Several unpublished studies have shown that this degradation need not occur by adaptively setting the gate value based on analysis of the separation of a particular sensor's tracks. Meanwhile, Figure 5 "zooms in" on the results shown in Figure 4 by showing the difference in average PCA for GNPL, what many regard as the leading algorithm for this class of problems, and that of the other algorithms. Thus, a positive delta PCA implies that GNPL made more correct associations on average. These results illustrate that the multistart heuristics are capable of achieving similar performance as GNPL.

The importance of using an adaptive gate value is underscored when looking at scenarios in which the number of common tracks n_C is low (i.e., track set overlap is 50% or less). In these scenarios, the single start local search method shows the best performance with respect to the probability of correct association metric. This behavior is due to the fixed gate value, which was set large and, therefore, reflects our preference to assign tracks in N_A with those in N_B as opposed to making a null assignment. This static gate value leads GNPL and the multistart local search methods to find local minima in which more assignments are made than really should be at a lower GNPM (negative log) likelihood than that of the local minimum at which the single start method terminates. Meanwhile, the local minimum found by the single start method corresponds to an assignment vector that makes, on average, a greater number of null assignments, and thus appears to be superior.

While difference in average PCA is not too pronounced across the five algorithms, Figure 6 shows the average run time (in log space) and reveals a more impressive finding. We see that when objects are closely-spaced, GNPL takes 50 to 100 times as long to find a solution of the same quality as the various multistart heuristics. Tracks for a single sensor need to be separated by at least 3 standard deviations in position error before GNPL returns a solution within a second. Meanwhile, the multistart heuristics solve any problem instance in roughly one- to two-tenths of a second.

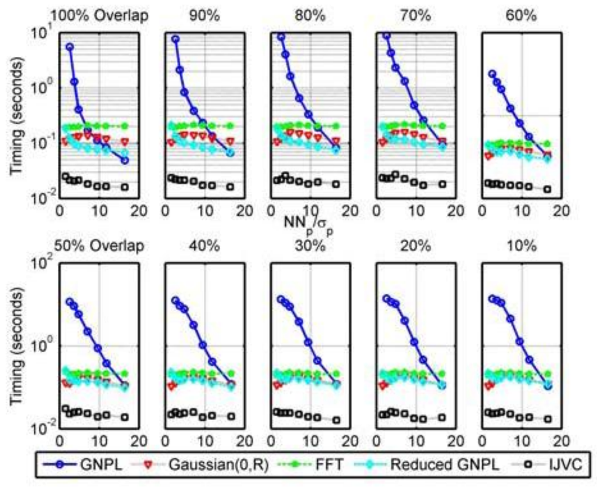

Figure 6 – Average CPU time (in seconds)

Figures 7 and 8 apply to a larger 20 on 100 scenario in which all sensor A tracks are seen by sensor B. Besides PCA, Figure 7 shows performance with respect to a metric we term "RV PCA." The missile defense community is often interested in the correct association of a single track, the so-called re-entry vehicle (RV) which carries a warhead. Like in many other test scenarios not shown here, the multistart heuristics have comparable performance with GNPL. We also see that algorithms 1 and 4, which use a branch-and-bound approach in some form, experience prolonged run times (100 seconds for GNPL and 10 seconds for Reduced GNPL with local search) for certain problem instances. These results add further evidence that branch-and-bound approaches are not robust enough to be used as standalone solvers for the GNPM problem in real-time applications.

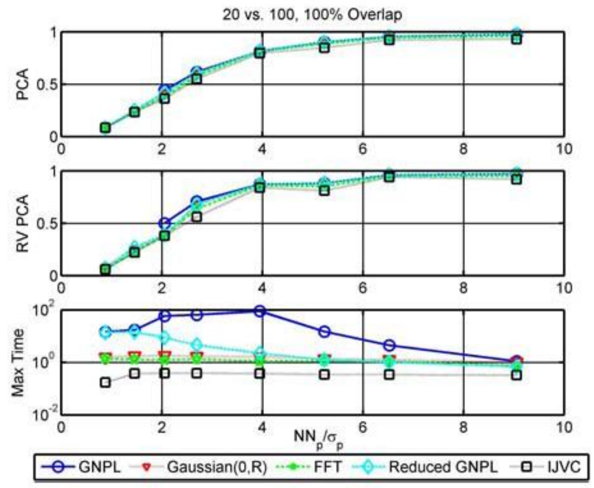

Figure 7 – Results for a 20 on 100 scenario

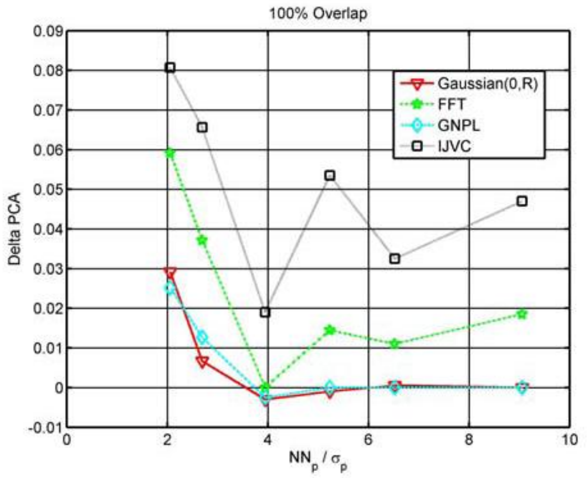

Figure 8 – Delta PCA for a 20 on 100 scenario

Figure 9 shows how the Gaussian(**0**,**R**) variant of the multistart local search heuristic scales as the number of tracks increases. Specifically, for 100 randomly generated problem instances in which track positions were drawn uniformly within a cube with a side length of 200 meters, $N = n_A = n_B = n_T$ tracks were created for $N = 20, 30, \ldots, 100$. Then, one or thirty initial biases were randomly drawn from a Gaussian(**0**,**R**) distribution and one or thirty solutions to the GNPM problem were found. The resulting graphic allows one to interpolate or extrapolate, in an approximate sense, how much time would be required to solve an instance of the GNPM problem given a certain number of biases and for a certain number of solutions.

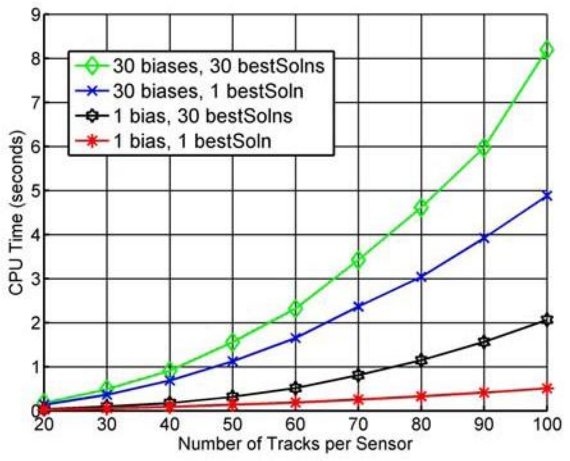

Figure 9 – Timing for N vs. N scenarios

5. Conclusions

Aligning and associating data from disparate sensor systems is a challenging, but nonetheless essential task for multisensor data fusion. Track-to-track association and bias estimation problems that can be formulated as a GNPM problem have several algorithms, both optimal and suboptimal, available for efficient solution. From a bird's eye view, the branch-and-bound method that we described "pursues" a global minimum by working in the solution space of all feasible assignments and perceives the bias estimate at a byproduct. In contrast, the multistart local search method pursues multiple local minima by explicitly using bias and assignment information in an iterative manner and, in so doing, makes the landscape of the GNPM (negative log) likelihood function (see Figure 2) the centerpiece of the algorithm. These two perspectives lead to different ways of understanding and attacking the GNPM problem.

When the number of objects is small and the scene density is low, a branch-and-bound algorithm guarantees an optimal solution to the GNPM problem in a reasonable amount of time. On the other hand, when the number of objects and scene density increase, we have shown that multistart local search heuristics exist that are competitive in terms of making the correct association but for significantly less computation time. It therefore seems reasonable to assume that real-time systems should have a hybrid design where optimality is guaranteed for small problem instances where computation time is not an issue, while a heuristic approach is taken when certain conditions on the scene density and the number of tracks are met.

Several extensions are already being investigated to capture greater realism:

(1) Elliptical covariances: In reality, covariance matrices typically have a pronounced "pancake" shape due to very precise measurement capabilities in the range dimension, and less precise capabilities in angle dimensions.

(2) Clustering: When the resolution of one sensor is known to be superior to that of another, it may be possible to reduce the number of tracks to associate by clustering together (into a single "meta-track") closely-spaced tracks of the high resolution sensor that the low resolution sensor would see as a single track. Algorithms for performing this clustering must take into account sensor-specific attributes.

(3) Feature/attribute information: Incorporating (possibly biased) feature information on tracks remains a topic of interest for improving association.

(4) Adaptive gating: The choice of an optimal gate value is still an open question. In fact, as described in [4], spurious observations, i.e., observations in N_A for which there is no corresponding match in N_B, have long hindered algorithms for solving generic point matching problems.

(5) Track ambiguity management: Once the K "best" bias-assignment hypotheses have been found, there is still a question of how to manage the tracks, or the uncertainty in the tracks, from a macroscopic system level.

From an optimization perspective, we believe it is only a matter of time before other mainstream metaheuristics, such as simulated annealing, genetic algorithms, and scatter search and tabu search, are applied to solve the GNPM problem. At the same time, there are several open questions that future research can address:

(1) Can one approximate the GNPM likelihood function in order to exploit standard optimization methods, e.g., convex, quadratic, or linear programming, for which commercial solvers are already available?

(2) Unlike a reduced branch-and-bound method, are there more efficient (i.e., polynomial-time) methods for finding tight lower bounds on the GNPM negative log likelihood function?

In sum, there are still a number of avenues that can be explored to improve track-to-track association in the face of sensor bias and other complicating factors.

APPENDIX: PROOF OF PRUNING CRITERION

We now prove that the quantity

$$h\left(\hat{\mathbf{b}}(\mathbf{p}), \mathbf{p}\right) + \min_{\mathbf{y} \in \Upsilon(k)} \sum_{i=k+1}^{n_A} \sum_{j=0}^{n_B} \begin{Bmatrix} \log\left(\left|\mathbf{S}_{ij}\right|\right) & \text{if } j > 0 \\ g & \text{if } j = 0 \end{Bmatrix} y_{ij}$$

from section 3 is a valid lower bound for a node with partial assignment $\mathbf{p}$ at depth k, for $k = 1, \ldots, n_A$. Consider any complete feasible assignment vector $\mathbf{j}$ whose first k components correspond to those of $\mathbf{p}$. Then,

$$\begin{aligned} NLL(\mathbf{b}, \mathbf{j}) &= \mathbf{b}^T \mathbf{R}^{-1} \mathbf{b} + \sum_{i=1}^{n_A} \begin{Bmatrix} d_{ij_i}^2(\mathbf{b}) + \log\left(\left|\mathbf{S}_{ij_i}\right|\right) & \text{if } j_i > 0 \\ g & \text{if } j_i = 0 \end{Bmatrix} \\ &\geq \sum_{i=1}^{k} \begin{Bmatrix} d_{ij_i}^2(\mathbf{b}) + \log\left(\left|\mathbf{S}_{ij_i}\right|\right) & \text{if } j_i > 0 \\ g & \text{if } j_i = 0 \end{Bmatrix} \\ &\qquad + \sum_{i=k+1}^{n_A} \begin{Bmatrix} d_{ij_i}^2(\mathbf{b}) + \log\left(\left|\mathbf{S}_{ij_i}\right|\right) & \text{if } j_i > 0 \\ g & \text{if } j_i = 0 \end{Bmatrix} \\ &\geq \sum_{i=1}^{k} \begin{Bmatrix} d_{ij_i}^2(\hat{\mathbf{b}}(\mathbf{p})) + \log\left(\left|\mathbf{S}_{ij_i}\right|\right) & \text{if } j_i > 0 \\ g & \text{if } j_i = 0 \end{Bmatrix} \\ &\qquad + \sum_{i=k+1}^{n_A} \begin{Bmatrix} \log\left(\left|\mathbf{S}_{ij_i}\right|\right) & \text{if } j_i > 0 \\ g & \text{if } j_i = 0 \end{Bmatrix} \\ &\geq h\left(\hat{\mathbf{b}}(\mathbf{p}), \mathbf{p}\right) + \min_{\mathbf{y} \in \Upsilon(k)} \sum_{i=k+1}^{n_A} \sum_{j=0}^{n_B} \begin{Bmatrix} \log\left(\left|\mathbf{S}_{ij}\right|\right) & \text{if } j > 0 \\ g & \text{if } j = 0 \end{Bmatrix} y_{ij} \end{aligned}$$

The first inequality follows since the term $\mathbf{b}^T \mathbf{R}^{-1} \mathbf{b}$ is always nonnegative. The second inequality follows from the definition of $\hat{\mathbf{b}}(\mathbf{p})$, the correspondence between $\mathbf{p}$ and $\mathbf{j}$, and the fact that $d_{ij}^2(\mathbf{b})$ is always nonnegative. The last inequality is due to the second term being a linear assignment problem over the remaining unassigned tracks in N_A and N_B.

ACKNOWLEDGEMENTS

The authors wish to thank Alan Chen, Fred Daum, Jim Huang, Dave Kingsland, Ron Kusner, and Mike Salpukas – all of Raytheon – for participating in stimulating discussions that enhanced the quality and presentation of this work.

References

[1] S.S. Blackman and N.D. Banh, "Track Association Using Correction for Bias and Missing Data," Proc. of SPIE Vol. 2235-529, 1994.

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

[3] L.G. Brown, "A Survey of Image Registration Techniques," ACM Computing Surveys, Vol. 24, No. 4, 1992.

[4] J. Denton and J.R. Beveridge, "An Algorithm for Projective Point Matching in the Presence of Spurious Points," Pattern Recognition 40, pp. 586-595, 2007.

[5] O.E. Drummond, D.A. Castañon, and M.S. Bellovin, "Comparison of 2-D Assignment Algorithms for Rectangular, Floating Point Cost Matrices," Proc. of SDI Panels on Tracking, No. 4, pp. 81-97, Dec. 1990.

[6] M.J. Hirsch, "GRASP-Based Heuristics for Continuous Global Optimization Problems," Ph.D. thesis, University of Florida at Gainesville, 2006.

[7] JAMA: A Java Matrix Package. Last time accessed: October 2007. http://math.nist.gov/javanumerics/jama/.

[8] R. Jonkers and A. Volgenant, "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems," Computing, Vol. 39, pp. 325-340, 1987.

[9] M. Levedahl, "An Explicit Pattern Matching Assignment Algorithm," Signal and Digital Processing of Small Targets (O.E. Drummond, Editor), Proc. of SPIE Vol. 4728, 2002.

[10] M.L. Miller, H.S. Stone, and I.J. Cox, "Optimizing Murty's Ranked Assignment Method," IEEE Transactions on Aerospace and Electronic Systems, Vol. 33, No. 3, July 1997.

[11] J.R. Moore and W.D. Blair, "Practical Aspects of Multisensor Tracking," in *Multitarget-Multisensor Tracking: Applications and Advances*, Vol. III (Y. Bar-Shalom and W.D. Blair, eds.) Norwood, MA: Artech House, 2000.

[12] K.G. Murty, "An Algorithm for Ranking All of the Assignments in Increasing Cost," Operations Research 16, pp. 682-687, 1968.

[13] G.L. Nemhauser and L.A. Wolsey, "Integer and Combinatorial Optimization," Wiley, 1988.

[14] K.R. Pattipati, R.L. Popp, and T. Kirubarajan, "Survey of Assignment Techniques for Multitarget Tracking," in *Multitarget-Multisensor Tracking: Applications and Advances*, Vol. III (Y. Bar-Shalom and W.D. Blair, eds.) Norwood, MA: Artech House, 2000.

[15] C.R. Pedersen, L.R. Nielsen, and K.A. Andersen, "An Algorithm for Ranking Assignments Using Reoptimization," Computers and Operations Research, 2007.

[16] L.D. Stone, M.L. Williams, and T.M. Tran, "Track-to-Track Association and Bias Removal," Signal and Data Processing of Small Targets, O.E. Drummond, Editor, Proceedings of SPIE Vol. 4728, pp. 315-329, 2002.

Biography

Dimitri Papageorgiou is a systems engineer at the Raytheon Company, where he has been heavily involved in projects related to sensor resource management, multi-target tracking, and data association. He is a recipient of a US National Science Foundation Graduate Research Fellowship in the field of Operations Research. He holds a BS in Mathematics from the University of North Carolina at Chapel Hill and an MS in Operations Research and Industrial Engineering from the University of Texas at Austin.

John-David Sergi is a senior systems engineer at Raytheon Integrated Defense Systems in Woburn, Massachusetts. He has been working at Raytheon for the last six years within the Center for Discrimination Algorithms. A large portion of his work has focused on tracking and sensor fusion as applied to the missile defense problem. He earned a bachelor's degree in electrical engineering from Tufts University in Medford, Massachusetts. He is currently pursuing a Masters degree on a part time basis.