Multi-Target Tracking via Mixed Integer Optimization

Dimitris Bertsimas, Zachary Saunders, and Shimrit Shtern

***Abstract*—The field of multi-target tracking faces two primary challenges: (i) data association and (ii) trajectory estimation. MTT problems are well researched with many algorithms solving these two problems separately, however few algorithms attempt to solve these simultaneously and even fewer utilize optimization. In this paper we introduce a new mixed integer optimization (MIO) model which solves the data association and trajectory estimation problems simultaneously by minimizing an easily interpretable global objective function. Furthermore, we propose a greedy heuristic which quickly finds good solutions. We extend both the heuristic and the MIO model to scenarios with missed detections and false alarms.**

***Index Terms*—optimization; multi-target tracking; data association; trajectory estimation; mixed integer optimization**

I. Introduction

MULTI-target tracking is the problem of estimation the state of multiple dynamic objects, referred to as *targets* over a fixed window of time. At various points of time within the window, the targets are observed in a *scan*, resulting in set of *detections*. From these detections, the multi-target tracking problem aims to extract information about target dynamics.

Solutions to this problem are sought across many civilian and military applications including but not limited to ballistic missile and aircraft defense, space applications, the movement of ships and ground troops, autonomous vehicles and robotics, and air traffic control. Each application has unique attributes and assumptions, and various algorithms have been developed for each. As a result, the field of multi-target tracking has expanded to numerous research venues, and there is a wide range of literature on the topic. A more complete overview of all MTT methods, including the classes of algorithms and their variants as well as additional methods not discussed in this paper, can be found in [1]. For a more exhaustive overview of estimation techniques, filtering, gating, and more please see [2] or [3].

The field of multi-target tracking faces two primary challenges: (i) data association and (ii) trajectory estimation. Given a set of sensor detections the data association problem consists of assigning the detections to a set of targets. Alternatively, this can be viewed as a labeling problem in which each detection needs to be labeled with a target identifier. The association problem is further complicated when sensors fail to report detections (missed detection) or incorrectly report detections (false alarm), resulting in ambiguity in the number of existing targets. The trajectory estimation problem consists of estimating the state space of a target (*i.e.*, position, velocity, acceleration, size, etc.) from the associated detections of the aforementioned assignment problem. Even when all of the associations are known, the estimation problem is challenging due to the presence of measurement noise. As can be seen, the two problems of data association and trajectory estimation are closely related and dependent on one another.

Some classical algorithms treat the data association and trajectory estimation problems separately using a combination of probabilistic approaches to determine data associations and filters to estimate trajectories. One such algorithm is the global nearest neighbor (GNN). The GNN algorithm is a naive 2-D assignment algorithm, which evaluates one scan of detections at a time, globally assigning the nearest detection at each scan [4]. Once the data association has been determined, the detections are often passed through one of numerous filters, most commonly a Kalman filter [5], which updates the trajectory estimates before the algorithm progresses forward to the next scan. This process repeats sequentially through each scan of data.

Modern algorithms in the field of multi-target tracking are most commonly statistical based, often relying on heavy probabilistic assumptions about the underlying target dynamics or detection process. The two most prevalent statistical algorithms in the field of multi-target tracking are the the Multiple Hypothesis Tracker (MHT) and the Joint Probability Data Association Filter (JPDAF) and their numerous variants and extensions. Both classes of algorithms attempt to solve the data association problem by generating a set of potential hypotheses, or possible detection-to-track assignments. Here a *track* is a set of labelled detections belonging to the same target. Probabilities are assigned to each hypothesis based on the likelihood of the trajectory's existence, and numerous approaches for accomplishing this task have been proposed.

The MHT, first proposed by Reid in [6], assigns likelihood values to hypotheses using a Bayesian MAP estimator, which requires assumptions on object dynamics. This algorithm is generally considered to be the modern standard for solving the data association problem. Many variants have been proposed for implementation which leverage techniques such as clustering, gating, hypothesis selection, hypothesis pruning, and merging of state estimates. Many of these methods are summarized by Blackman in [7].

While the MHT has seen various forms of success, it faces several key challenges. Namely, the curse of dimensionality and complexity. The number of possible hypotheses grows exponentially with the number of potential tracks and the number of scans. Consequently, it is considered intractable for large scenarios. Moreover, the MHT might require extensive tuning and thus may be difficult to implement in practice, in addition to being computationally expensive. For these reasons

it is generally considered to be one of the most complex MTT algorithms.

A Probability Data Association (PDA) takes a Bayesian approach to solving the data association problem by finding detection-to-target assignment probabilities via a posterior PDF, which again requires heavy assumptions on object dynamics and the detection process. In similar fashion, a Joint PDA (JPDA) assigns probabilities that are computed *jointly* across all targets. The JPDAF is an algorithm which implements the JPDA along with filters and estimation methods as discussed previously. [2]

A limited number of optimization based algorithms have been applied to solve the MTT problem, most of which attempt to solve by mapping the measurement set onto a trellis and seek the optimal measurement association sequence. Some examples include the Multi-Target Viterbi[8] and an extension in [9] which formulates [8] as a network flow, reducing the solve time from exponential to polynomial. Others have suggested adaptations which allow this approach to be used similar to MHT methods by outputting a single best set of K tracks, or a list of L best sets of k tracks [10].

Compared to the number of statistical based algorithms in the MTT literature, optimization based algorithms are relatively lacking. In fact, most occurrences of optimization in the MTT literature propose the use of optimization to leverage statistical algorithms, in particular the MHT. For example, Integer optimization has been used to improve MHT hypothesis selection by solving an assignment problem which chooses the best hypothesis, but only after costs have been assigned (statistically based) and hypotheses have been pruned [11]. Somewhat similarly, linear optimization has also been used to assist in the hypothesis selection process for the MHT [12]. Still, other attempts aim to improve the MHT hypothesis selection process via Lagrangian relaxation [13].

More recently, Andriyenko and Schindler have proposed formulating the MTT problem as a minimization of a continuous energy in [14] and then again as a minimization of discrete-continuous energy in [15]. These algorithms aim to more accurately represent the nature of the problem, but sacrifice interpretability for complexity in the process. Rather than formulating the problem to lend it easily to traditional global optimization methods, the authors intend to leverage the use of optimization techniques to find strong local minima of their proposed energy objective, and they achieve strong results in doing so. However, this approach calls for the use of several parameters that must be tuned and few recommendations are provided for how to go about such a tuning process. Additionally, these methods require initialization heuristics to begin the solving process, which is in itself complicated to implement and is not directly connected to the optimization problem solved.

In this paper we propose the use of mixed integer optimization (MIO) to formulate and solve the multi-target tracking problem. Although MIOs are generally thought to be intractable (NP-Hard), in many practical cases near optimal solutions and even optimal solutions to these problems can be obtained in reasonable time [16]. This can be attributed to the fact that MIO solvers have seen significant performance improvements in recent years due to advancements in both methodology and hardware. The development of new heuristic methods, discoveries in cutting plane theory, and improved linear optimization methods have all contributed to improvements in performance [17]. Modern solvers such as Gurobi and CPLEX have been shown to perform extremely well on benchmark tests. In the past six years alone, Gurobi has seen performance improvements by a factor of 48.7 [18]. CPLEX saw improvements by a factor of 29,000 from 1991 to 2007 [19]. From 1994 to 2014, the growth of supercomputing power as recorded by the TOP500 list has improved by a factor of 567839 [20]. Thus, the total combined effective improvement of software and hardware advancements is on the scale of 800 billion times in the past 25 years.

The literature is also lacking in performance metrics for the evaluation of MTT algorithms. There is no standard method of measuring scenario complexity or algorithm performance as a function of this complexity. In many cases only the sensor's detection noise is taken into account and other factors such as target density is negated. Recent work [21] proposes a mathematically rigorous performance metric for measuring the distance between ground truth and estimated track, but there is not much attention given to the complexity of generated scenarios. In this paper we also suggest measures of complexity and performance which are related to the ones suggested in [21] but we show the value in relating a complexity measure to performance measures, namely that it allows you to evaluate the data association and trajectory estimation problems separately. We evaluate the methods suggested in this paper using these complexity and performance measures on two simulated experiments.

The main contributions of this paper are as follows:

(i) We introduce a simple interpretable MIO model which solves the data association and trajectory estimation problems simultaneously for a sensor with no detection ambiguity. The model does not require to tuning of parameters. This MIO is tractable, in the sense that it can be solved to optimality or near optimality in a reasonable amount of time, for the considered applications.

(ii) We propose a heuristic, motivated by the optimization problem, which gives us feasible solutions to this problem and show how it can be used as warm start to the MIO in order to improve the quality of the solutions obtained as well as the running time.

(iii) We extend this basic MIO model and corresponding heuristic initialization algorithm for the case of detection ambiguity, i.e., the case where there are both missed detections and false alarms, keeping interpretability while only adding two tunable parameters, as well as provide general guidelines as to how tune this parameters.

(iv) We present a new measure of complexity for the data association problem, and show how it aids in scenario generation. We also discuss a simplified measure of performance for the trajectory estimation problem.

The paper structure is as follows. We begin with a description of the MTT problem as we wish to model it in Section II. In Section III we develop a simple MIO formulation

for a sensor with no detection ambiguity and extend it to a generalized formulation. Following is a discussion on a proposed heuristic in Section IV. Then we propose extensions for both the MIO formulation and the heuristic to a sensor with detection ambiguity in Section V. Metrics for measuring scenario complexity and algorithm performance are proposed in Section VI. Experimental simulations are outlined in Section VII. A summary of significant computational results are discussed in Section VIII. Finally, conclusions and future work are discussed in Section IX.

General Notations: Unless specified otherwise, $\|\cdot\|$ is used to indicate the L1 norm, and $|\cdot|$ refers to element wise absolute value.

II. Problem Description

In this paper, we restrict our exploration of the MTT problem to the automatic tracking of multiple, independent point targets using a single sensor. A *target* is the object of interest. A point target's only identifiable attributes are its state space, which we restrict to position and velocity. The state space fully defines the field of *trajectories*, or paths along which targets travel. A *detection* is collected from each target at sequential scans. Detections are subject to noise. We treat two general scenarios: with and without detection ambiguity.

When there is no detection ambiguity, the sensor produces exactly one detection for each target at each time, and there is no other source of detections. Therefore, the number of detections at each point in time is the same as the number of targets, and the data association problem at each point in time is equivalent to a simple assignment problem. Our basic optimization model, presented in section [add reference] will this with this problem.

Detection ambiguity refers to the more complex case where the sensor generates both false alarms and missed detections. A *False Alarm* occurs when a detection is collected when no target exists. This could be the result of measurement error or difficulties in signal processing. A *Missed Detection* occurs when a data point is not collected at a given time when a target actually exists. Therefore, the number of detections at each point in time may be either higher or lower than the actual number of targets, and each detection can be classified in either of these categories in addition to assigning targets to trajectories as before. In section [add reference] we will extend the formulation of basic model to a robust formulation dealing with this ambiguity, and refer to it as the robust MIO model.

Throughout the paper we make the following assumptions:

Assumption 1. *(i) All targets have constant velocity. i.e., Targets do not maneuver and no outside forces act on them.*
(ii) Each target's dynamics are independent of one another.
(iii) The number of targets remains constant throughout the window of observation, i.e., there is no birth/death of targets.
(iv) Each target produces at most one detection per scan.
(v) The detection errors are independent of one another.

Notation: We observe P targets over a fixed time window over which T scans are collected. Scans occur at a fixed rate, usually of about 1Hz, such that the set of scans is denoted by $\{t_1, t_2, ..., T\}$. The i^{th} detection of the t^{th} scan is indicated by x_{it}, such that a scan of data at time *t* is the unordered set of detections $\mathcal{X}_t = \{x_{1t}, x_{2,t}, ..., x_{P,t}\}$. The data for the problem is the ordered set of scans $\boldsymbol{\mathcal{X}} = \{\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_T\}$. The state space of target trajectories is paramatarized by a true initial position α_j^{true} and a true constant velocity β_j^{true}. Therefore, the true position $\bar{x}_{jt}$ of trajectory j at scan t is given by:

$$\bar{x}_{jt} = \alpha_j^{true} + \beta_j^{true} t \tag{1}$$

III. Basic MIO Model

In this section, we deal with the case of no detection ambiguity. Therefore, we add the following, more restrictive assumptions, to those presented in Assumption 1

Assumption 2. *(i) The sensor generates exactly one detection for each target at each time (no missed detections).*
(ii) The sensor does not generate any additional detections (no false alarms).

We begin constructing our MIO model by defining decision variables that represent the desired detection to target associations and target estimated trajectories. Next, using these decision variables, we develop an objective function which mathematically quantifies the value of the model decisions, in this case as a measure of distance of the estimated trajectories from the associated detections. Finally, we restrict these variables using constraints that force the model to find solutions that are feasible for the MTT problem. The model is developed step by step in the coming sections before the full model is presented.

A. Decision Variables

The data association and trajectory estimation problems require unique decision variables. Because these two problems lie in different domains, the variables we use to represent these decisions also differ. First, we introduce *continuous* decision variables $\alpha_j \in \mathbb{R}^n$ and $\beta_j \in \mathbb{R}^n$ to represent the estimated initial position and velocity of each trajectory *j*. In our interpretation of the MTT problem we allow the trajectory parameters to lie anywhere in the real-continuous domain. For the data estimation problem, we wish to assign detections to trajectories, a naturally discrete problem. Therefore, we introduce binary decision variables y_{itj} to indicate whether detection x_{it} is assigned to trajectory *j* or not:

$$y_{itj} = \begin{cases} 1, & \text{if detection } x_{it} \text{ is assigned to trajectory } j, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

B. Objective Function

Next, we would like to develop a function which accurately scores the quality of a feasible solution. An ideal objective function would jointly provide a single quantifiable measure

of goodness for both the data association and trajectory estimation problems. Therefore we want the objective to take into account the assignments of detections in addition to the estimated trajectory determined by those assignments.

In terms of our established decision variables, the estimated position of the linear trajectory *j* at time *t* is given by:

$$\hat{x}_{jt} = \alpha_j + \beta_j t \tag{3}$$

For the trajectory estimation problem we wish to minimize the distance between x_{it} and $\hat{x}_{jt}$. In other words, for detections x_{it} assigned to trajectory j, we wish to minimize $\|x_{it} - \hat{x}_{jt}\|$ for some norm. The two natural norms to consider here are the L1 and L2 norms. The L1 norm has the advantage that it can be reformulated using linear optimization (through the addition of continuous variables and constraints), and it is well known to be more robust to outliers. Additionally, existing algorithms for MIO are more well developed for linear rather than quadratic optimization. However, the L2 norm square form (RSS) has the advantage that it can be quickly computed using a matrix formulation, making it more predisposed to a heuristic. This concept will be discussed further in section IV.

Substituting (3) for $\hat{x}_{jt}$ we arrive at our objective function:

$$\underset{\alpha_j,\beta_j}{\text{minimize:}} \sum_{(i,j)\in\mathcal{A}} \sum_{t=1}^{T} \|x_{it} - \alpha_j - \beta_j t\| \tag{4}$$

where $\mathcal{A}$ is the set of pairs that indicate the assignment of detection *i* to trajectory *j*.

For the data association problem, we wish to only penalize the objective function when detection x_{it} has been assigned to trajectory *j*, which occurs when $y_{itj} = 1$. An easy method to enforce this using our established variables would be to construct an interaction term like in (5) below.

$$\underset{y_{itj},\alpha_j,\beta_j}{\text{minimize:}} \sum_{i=1}^{P} \sum_{t=1}^{T} |y_{itj} x_{it} - \alpha_j - \beta_j t\| \tag{5}$$

We now show that this objective can be converted to a linear program in the case of the L1 norm by introducing continuous variables θ_{jt} and the following additional constraints.

$$y_{itj} x_{it} - \alpha_j - \beta_j t \le \theta_{jt} \qquad \forall i, j, t \tag{6}$$
$$-(y_{itj} x_{it} - \alpha_j - \beta_j t) \ge \theta_{jt} \qquad \forall i, j, t \tag{7}$$

The resulting objective function for the case of the L1 norm would then be:

$$\underset{\theta_{jt}}{\text{minimize:}} \sum_{j=1}^{P} \sum_{t=1}^{T} \theta_{jt} \tag{8}$$

where e is the vector of ones.

For the case of the L2 norm, the objective function would be:

$$\underset{\theta_{jt}}{\text{minimize:}} \sum_{j=1}^{P} \sum_{t=1}^{T} \|\theta_{jt}\|_2^2 \tag{9}$$

C. Constraints

For each scan, each detection x_{it} must be assigned to exactly one target *j*:

$$\sum_{j=1}^{P} y_{itj} = 1 \qquad \forall i, t \tag{10}$$

Similarly, for each scan, each target must be assigned exactly one detection:

$$\sum_{i=1}^{P} y_{itj} = 1 \qquad \forall j, t \tag{11}$$

D. Simple Formulation

Combining all of these elements together, we arrive at the following MIO model:

$$\begin{aligned}
\underset{\theta_{jt}}{\text{minimize:}} \quad & \sum_{j=1}^{P} \sum_{t=1}^{T} \theta_{jt} \\
\text{subject to:} \quad & \sum_{j=1}^{P} y_{itj} = 1 \qquad \forall i, t \\
& \sum_{i=1}^{P} y_{itj} = 1 \qquad \forall j, t \\
& y_{itj} x_{it} - \alpha_j - \beta_j t \le \theta_{jt} \qquad \forall i, j, t \\
& -(y_{itj} x_{it} - \alpha_j - \beta_j t) \ge \theta_{jt} \qquad \forall i, j, t \\
& y_{itj} \in \{0, 1\} \quad \forall i, t, j \\
& \alpha_j \in \mathbb{R}^n \quad \forall j, \quad \beta_j \in \mathbb{R}^n \quad \forall j, \quad z_{jt} \in \mathbb{R}^n \quad \forall j, t
\end{aligned}$$

E. Generalized Formulation

The previous formulation, although simple and easily interpretable, has the disadvantages of being (*i*) dense and (*iI*) ill suited for extension to detection ambiguity. Therefore, next, we present a generalized formulation which can be linearized through the introduction of additional binary decision variables. Alternatively, we can create a new variable z_{jt} which takes on the value x_{it} when $y_{ijt} = 1$ and some arbitrary number when $y_{itj} = 0$. Using this method we must adjust the objective function below.

$$\underset{z_{jt},\alpha_j,\beta_j}{\text{minimize:}} \sum_{j=1}^{P} \sum_{t=1}^{T} \|z_{jt} - \alpha_j - \beta_j t\| \tag{12}$$

This objective can then be linearized by again introducing θjt and similar constraints as follows.

$$\underset{\theta_{jt}}{\text{minimize:}} \sum_{j=1}^{P} \sum_{t=1}^{T} \theta_{jt} \tag{13}$$

$$z_{jt} - \alpha_j - \beta_j t \le \theta_{jt} \qquad \forall i, j, t \tag{14}$$
$$-(z_{jt} - \alpha_j - \beta_j t) \ge \theta_{jt} \qquad \forall i, j, t \tag{15}$$

Furthermore, we must ensure that the decision variable z_{jt} will only take on the value of x_{it} in the objective function if

x_{it} is assigned to target *j* ($y_{itj} = 1$). We enforce this effect using the following constraint:

$$M_t(1 - y_{itj}) \geq |z_{jt} - x_{it}y_{itj}| \qquad \forall i, t, j \tag{16}$$
$$\tag{17}$$

where $M_t = \max_j |x_{it}|$ for each scan. Furthermore, we can write this equivalently as a linear optimization problem by using the following set of two linear constraints:

$$x_{it}y_{itj} + M_t(1 - y_{itj}) \geq z_{jt} \qquad \forall i, t, j \tag{18}$$
$$x_{it}y_{itj} - M_t(1 - y_{itj}) \leq z_{jt} \qquad \forall i, t, j \tag{19}$$

Combining all of these elements together, we arrive at the following generalized MIO model:

$$\begin{aligned}
\underset{\theta_{jt}}{\text{minimize:}} \quad & \sum_{j=1}^{P}\sum_{t=1}^{T}\theta_{jt} \\
\text{subject to:} \quad & \sum_{j=1}^{P} y_{itj} = 1 \qquad \forall i, t \\
& \sum_{i=1}^{P} y_{itj} = 1 \qquad \forall j, t \\
& x_{it}y_{itj} + M_t(1 - y_{itj}) \geq z_{jt} \qquad \forall i, t, j \\
& x_{it}y_{itj} - M_t(1 - y_{itj}) \leq z_{jt} \qquad \forall i, t, j \\
& z_{jt} - \alpha_j - \beta_j t \leq \theta_{jt} \qquad \forall i, j, t \\
& -(z_{jt} - \alpha_j - \beta_j t) \geq \theta_{jt} \qquad \forall i, j, t \\
& y_{itj} \in \{0, 1\} \quad \forall i, t, j \\
& \alpha_j \in \mathbb{R}^n \quad \forall j, \quad \beta_j \in \mathbb{R}^n \quad \forall j, \quad z_{jt} \in \mathbb{R}^n \quad \forall j, t
\end{aligned}$$

IV. Heuristic

Next, we present a detailed description of a heuristic which finds good feasible solutions. These solutions can be used as a warm start to the MIO, providing a performance boost to the MIO. The heuristic leverages the power of randomization local search methods to find locally optimal solutions. Although the heuristic takes a local search approach, we hypothesize that it will discover near optimal solutions, provide that it is initialized with enough random starting points.

An important distinction to discuss here is the difference in objective functions. As discussed before the two natural choices are the L1 and L2 norms. For the heuristic, we desire an objective which can be calculated efficiently. Therefore, in this case the L2 norm square (RSS) is the preferred choice because it can be calculated quickly using matrix algebra. [22] shows how a design matrix M can be using to quickly compute the RSS.

The algorithm initializes by randomizing a solution which satisfies equations 6 and 7. The initial parameters α_j and β_j are calculated as well as the objective score, RSS^0. In swap k for scan t choose $i, l \in \{1, \ldots, P\}$ detections and $j, m \in \{1, \ldots, P\}$ targets such that $y^k_{itj} = 1$ and $y^k_{ltm} = 1$. Switch the detection association so that $y^{k+1}_{ltj} = 1$ and $y^{k+1}_{itm} = 1$. Compute $\alpha_j, \beta_j, \alpha_m, \beta_m$, and RSS^k. If the objective score improves, the swap is kept, otherwise it is rejected. The algorithm then advances to the next scan where the same process is repeated, and it terminates once it makes a single pass through all scans without accepting a single switch. As we will see in the computational results section, Algorithm 1 runs very efficiently, providing high quality global solutions very quickly. Furthermore, this algorithm can be parallelized by running partitions of the N starting points on separate cores, leading to even greater performance advantages. A proposed pseudocode for the heuristic is provided below in Algorithm 1.

Algorithm 1 Randomized local search with heuristic swaps

Input: $\boldsymbol{\mathcal{X}}$, P, T, $\boldsymbol{M}$
Output: RSS

Initialization : Assign random initial assignments for y^0_{itj}
Calculate $\alpha_j, \beta_j \quad \forall j$
Calculate RSS^0
swapped $\leftarrow true$
$k \leftarrow 1$
while swapped **do**
  swapped $\leftarrow false$
  for t in $\{t_1, t_2, ..., T\}$ **do**
    Randomly choose $j, m \in \{1, \ldots, P\}$
    Find i, l such that $y^{k-1}_{itm} \leftarrow 1$ and $y^{k-1}_{ltj} \leftarrow 1$
    Swap such that $y^k_{itj} \leftarrow 1$ and $y^k_{ltm} \leftarrow 1$
    Calculate $RSS^k, \alpha_j, \beta_j, \alpha_m, \beta_m$
    if ($RSS^k \geq RSS^{k-1}$) **then**
      $y^k \leftarrow y^{k-1}$
    else
      swapped $\leftarrow true$
    end if
  end for
  $k \leftarrow k + 1$
end while
return RSS^k, y^k_{itj}

V. Robust MIO Model

In this section we treat the case of detection ambiguity. The key difference is that now the number of targets is unknown, and this becomes a third problem which we wish to solve in addition to the data association and trajectory estimation problems which remain once the number of targets has been determined. Since in this case both missed detections and false alarms are present the number of targets is unknown and we may no longer have the same number of detections at each scan. Therefore, we must introduce additional notation for this scenario. We let n_t represent the number of detections at time *t*. We can then identify the fewest and largest number of detections in a scan with $N_0 = \min_t n_t$ and $N_1 = \max_t n_t$, respectively.

Specifically, in this case we replace Assumption 2 by the following less restrictive assumptions.

Assumption 3. *(i) The sensor does not generate a detection for any target for any time with probability P_d which is constant and independent between targets and scans.*

(ii) At each point in time the sensor generates false alarms according to a Poisson distribution with rate λ_{FA}, which are located uniformally in the space.

(iii) The number of true targets P satisfies $N_0 \leq P \leq N_1$.

We first show that this problem can be solved by dividing it into a subset of simpler problems. We present a MIO formulation that assumes a fixed number of targets. This formulation allows us to leverage the power of parallelization to solve the problem by solving each subproblem separately. The results can then be gathered and compared to find the globally optimal solution. For completeness we also present a formulation which solves the original problem without the need for multiple parallelized MIOs.

A. Fixed Number of Targets (P)

If we first assume that the number of targets is fixed, we can more easily adapt the generalized formulation presented in Section III to handle the addition of false alarms and missed detections. This simple adaptation requires the introduction of two additional variable types and minimal constraint changes. We can then run these formulations for each possible value of fixed number of targets *P* across the range of N_0 to N_1 and choose the solution with the best objective overall. Furthermore, this is an advantageous strategy because each independent experiment can be run in parallel.

1) Decision Variables: We first introduce new binary decision variables F_{it} to indicate whether or not a detection x_{it} is a false alarm.

$$F_{it} = \begin{cases} 1, & \text{if detection } i \text{ at time } t \text{ is a False Alarm,} \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, we introduce binary decision variables M_{jt} to indicate whether or not an *existing* trajectory *j* has a missed detection at time *t*.

$$M_{jt} = \begin{cases} 1, & \text{if detection for trajectory } j \\ & \text{at time } t \text{ is a Missed Detection,} \\ 0, & \text{otherwise.} \end{cases}$$

2) Constraints: All detections must either be assigned to a trajectory *j* or a false alarm.

$$\sum_{j=1}^{P} y_{itj} + F_{it} = 1 \qquad \forall i, t \tag{20}$$

All trajectories *j* must either be assigned a detection or a missed detection.

$$\sum_{i=1}^{n_t} y_{itj} + M_{jt} = 1 \qquad \forall j, t \tag{21}$$

The sum of all false alarms is TF, and similarly the sum of all missed detections is TM.

$$\sum_{i=1}^{n_t} \sum_{t=1}^{T} F_{it} = TF \tag{22}$$

$$\sum_{j=1}^{P} \sum_{t=1}^{T} M_{jt} = TM \tag{23}$$

3) Objective Function: We can easily extend (12) to account for false alarms and missed detections by introducing penalties ψ_0 (ϕ_0, respectively) for each missed detection (false alarm, respectively). Such an objective form would take the form of:

$$\underset{z_{jt},\alpha_j,\beta_j,TF,TM}{\text{minimize:}} \sum_{j=1}^{P} \sum_{t=1}^{T} \theta_{jt} + \psi_o TF + \phi_0 TM \tag{24}$$

which can be linearized in the same manner as (13).

4) Formulation 2:

$$\begin{aligned}
\underset{\psi_{jt}}{\text{minimize:}} \quad & \sum_{j=1}^{P} \sum_{t=1}^{T} \psi_{jt} + \theta_o TF + \phi_0 TM \\
\text{subject to:} \quad & \sum_{j=1}^{P} y_{itj} + F_{it} = 1 \qquad \forall i, t \\
& \sum_{i=1}^{n_t} y_{itj} + M_{jt} = 1 \qquad \forall j, t \\
& \sum_{i=1}^{n_t} \sum_{t=1}^{T} F_{it} = TF \\
& \sum_{j=1}^{P} \sum_{t=1}^{T} M_{jt} = TM \\
& x_{it} y_{itj} + M_t (1 - y_{itj}) \geq z_{jt} \qquad \forall i, t, j \\
& x_{it} y_{itj} - M_t (1 - y_{itj}) \leq z_{jt} \qquad \forall i, t, j \\
& z_{jt} - \alpha_j - \beta_j t \leq \psi_{jt} \qquad \forall j, t \\
& -(z_{jt} - \alpha_j - \beta_j t) \leq \psi_{jt} \qquad \forall j, t \\
& y_{itj} \in \{0, 1\} \quad \forall i, t, j \\
& \alpha_j \in \mathbb{R}^n, \quad \beta_j \in \mathbb{R}^n \quad \forall j \\
& z_{jt} \in \mathbb{R}^n, \quad \forall j, t
\end{aligned}$$

B. Number of Targets as a Decision Variable

In the previous section, we assumed we knew the number of targets. In this section, the number of targets is determined via optimization.

1) Decision Variables: Toward this goal, we introduce a new binary decision variable w_j to indicate whether or not trajectory *j* corresponds to an existing target.

$$w_j = \begin{cases} 1, & \text{if trajectory } j \text{ exists,} \\ 0, & \text{otherwise.} \end{cases}$$

2) Constraints: Most constraints remain similar to their original counterparts, except now we must account for the possibility that some trajectories may not exist. Therefore, where before we summed over *P*, we will now be summing over N_1. This affects two constraints.

All detections must either be assigned to a trajectory *j* or a false alarm. This can be implemented similarly to (20),

except now we sum over N_1 because the number of targets is unknown but limited by N_1.

$$\sum_{j=1}^{N_1} y_{itj} + F_{it} = 1 \qquad \forall i, t \tag{25}$$

Similarly, (23) must be adjusted to sum over the maximal number of targets allowed N_1 .

$$\sum_{j=1}^{N_1} \sum_{t=1}^{T} M_{jt} = TM \tag{26}$$

All *existing* trajectories must either be assigned a detection or a missed detection.

$$\sum_{i=1}^{n_t} y_{itj} + M_{jt} = w_j \qquad \forall j, t \tag{27}$$

We restrict α_j and β_j to be zero if trajectory *j* does not exist. This ensures only existing trajectories are penalized in the objective function.

$$|\alpha_j| + |\beta_j| \leq M_0 w_j \qquad \forall j \tag{28}$$

Since $N_0 \leq P \leq N_1$, we can set $w_j = 1$ for all $j = 1, \ldots, N_0$, which leaves us with only $N_1 - N_0$ additional binary variables. We simply need the additional constraint

$$w_{N_0+1} \geq ... \geq w_{N_1} \tag{29}$$

which guarantees a unique w solution. Furthermore, we can replace (27) with the following two constraints:

$$\sum_{i=1}^{n_t} y_{itj} + M_{jt} = 1 \qquad \forall j = 1, ..., N_0, t \tag{30}$$

$$\sum_{i=1}^{n_t} y_{itj} + M_{jt} = wj \qquad \forall j = N_0, ..., N_1, t \tag{31}$$

3) Formulation 3: Incorporating these additional variables and constraints, we arrive at the following complete alternative formulation.

$$\begin{aligned}
\underset{y_{itj}, \alpha_j, \beta_j, F_{it}, M_{jt}}{\text{minimize:}} \quad & \sum_{j=1}^{N_1} \sum_{t=1}^{T} |z_{jt} - \alpha_j - \beta_j t| + \theta_o TF + \phi_0 TM \\
\text{subject to:} \quad & \sum_{j=1}^{N_1} y_{itj} + F_{it} = 1 \qquad \forall i, t \\
& \sum_{i=1}^{n_t} y_{itj} + M_{jt} = 1 \qquad \forall j = 1, ..., N_0, t \\
& \sum_{i=1}^{n_t} y_{itj} + M_{jt} = wj \qquad \forall j = N_0, ..., N_1, t \\
& \sum_{i=1}^{n_t} \sum_{t=1}^{T} F_{it} = TF \\
& \sum_{j=1}^{N_1} \sum_{t=1}^{T} M_{jt} = TM \\
& w_{N_0+1} \geq ... \geq w_{N_1} \\
& |\alpha_j| + |\beta_j| \leq M_0 w_j \qquad \forall j \\
& x_{it} y_{itj} + M_1 (1 - y_{itj}) \geq z_{jt} \qquad \forall i, t, j \\
& x_{it} y_{itj} - M_1 (1 - y_{itj}) \leq z_{jt} \qquad \forall i, t, j \\
& y_{itj} \in \{0, 1\} \quad \forall i, t, j \\
& \alpha_j \in \mathbb{R}^n, \quad \beta_j \in \mathbb{R}^n, \quad w_j \in \mathbb{R}^n \quad \forall j \\
& z_{jt} \in \mathbb{R}^n, \quad \forall j, t
\end{aligned}$$

C. Robust Extension to Algorithm 1

The heuristic for the scenario with ambiguity follows similarly from the heuristic developed under the scenario without ambiguity. The main difference is that now the options for making switches must include false alarms and missed detections. Therefore, the framework of the new algorithm is the same as for Algorithm 1, but the new variant of the heuristic randomly chooses from the following options:

1) Switch detection assignments between two existing targets.
2) Switch the detection assignment of an existing target with a false alarm.
3) Switch the detection assignment of an existing target with a missed detection for a different existing target.
4) Move the detection assignment of an existing target to a false alarm and replace it with a missed detection.
5) Move a false alarm into the location of a missed detection for an existing target.

We refer to this robust extension to Algorithm 1 as Algorithm 2. Similar to Algorithm 1, this robust extension will accept the switch/move if the objective score improves, and reject the switch/move otherwise. Algorithm 2 terminates under the same conditions as Algorithm 1. We expect Algorithm 2 to run slightly slower due to the increase in potential combinations of solutions.

VI. Scenario Complexity & Performance Metrics

There does not exist a unified approach for measuring scenario complexity as stated by [1] nor does there exist clear

measures of performance for each of the trajectory estimation and data association problems. In this paper, we argue that the data association problem has a natural performance metric but lacks a measure of complexity, while the trajectory estimation problem has a natural measure of complexity but lacks a clear performance metric.

In the case of the data association problem, the preferred performance metric often used in practice is % accuracy *i.e.*, the number of correct detection assignments out of the number of possible correct assignments. For the case without sensor ambiguity, the number of possible assignments is simply the total number of detections, or equivalently, the number of targets multiplied by the number of scans.

$$Accuracy = \frac{\text{\# correct assignments}}{\text{Total \# of detections}} = \frac{\text{\# correct assignments}}{PT} \tag{32}$$

In the case of sensor ambiguity, however, the number of possible correct assignments requires a deeper explanation. To develop a better understanding, we consider our goal, which is to correctly assign detections to targets and identify both false alarms and missed detections. With this in mind, we define the number of possible correct assignments as the number of targets multiplied by the number of scans plus the number of false alarms

$$Accuracy = \frac{\text{\# correct assignments}}{PT + \text{\# False Alarms}}. \tag{33}$$

Whereas accuracy serves as a good measure of performance for data association, there does not exist a corresponding measure of complexity which comparatively measures the difficulty of the data association problem. We argue that σ alone is not the best measure of difficulty for the data association problem. For example, a scenario with very close target trajectories may not actually be difficult to ascertain data associations even for small σ values, and similarly with high enough σ values even widely spaced targets could be difficult to differentiate. Therefore, we introduce a metric ρ to quantify this complexity. For ease of notation in developing this metric we first define D_{ijt} as the distance between one true trajectory *i* and another true trajectory *j*.

$$D_{ijt} = \|\alpha_i^{true} + \beta_i^{true} t - \alpha_j^{true} + \beta_j^{true} t\| \tag{34}$$

Additionally, we define a variable c_{ijt} that will take the value of 1 if the distance between trajectory *i* and trajectory *j* is greater than some constant. We propose the use of 2σ, since it is hard to distinguish detections which lie between target trajectories closer that.

$$c_{ijt} = \begin{cases} 1, & \text{if } D_{ijt} > 2\sigma, \\ 0, & \text{otherwise.} \end{cases}$$

Then the difficulty of a scenario in the sphere of the data association problem is quantified by the complexity measure ρ, which is the proportion of detection pairs that fall within a closely defined proximity to each other.

$$\rho = \frac{\sum_{t=1}^{T} \sum_{i<j} c_{ijt}}{\binom{P}{2} T} \tag{35}$$

This metric has several desirable attributes. First and foremost, it falls within the range of $[0, 1]$, identical to the range of accuracy, making it easily comparable. Secondly, it is easy to understand and interpret. Higher values of ρ indicate easier scenarios because fewer targets are within close proximity for a shorter amount of time, and vice versa. Finally, as we have defined it, ρ has an inverse relationship with σ, which means that it serves as a connection between scenario generation and performance measuring processes. While σ can be used more naturally for scenario generation, where it is useful as a parameter for signal noise, ρ can be calculated after the fact and used to quantify the difficulty of the scenario as it pertains to the data association problem.

In the case of the trajectory estimation problem, the preferred complexity metric often used in practice is σ. Increasing the signal noise may often lead to stronger bias in the trajectory estimation, especially in scenarios with fewer scans, and results in a deteriorated quality of the estimation. Therefore, we believe that σ is the correct metric for use in measuring the difficulty of the trajectory estimation problem.

However, establishing a performance metric for the trajectory estimation problem is necessary. We choose to implement a metric which captures the core goal of the trajectory estimation problem, that is to estimate a trajectory as close as possible to the true ground track.

$$\delta = \frac{\sum_{t=1}^{T} \sum_{j=1}^{P} \|\bar{x}_{jt} - \hat{x}_{jt}\|}{PT} \tag{36}$$

We match the true trajectories to the estimated trajectories using a one-to-one assignment problem which can be formulated using linear optimization. Lower values of δ correspond to higher performance because the distance between the estimated and true ground trajectories is smaller.

In Section , we will see how these measures of complexity and performance are useful in quantifying the strengths and weaknesses of our methods.

VII. Experimental Simulations

There does not exist among the literature a clearly defined comprehensive set of simulation scenarios as pointed out by [1]. However, this work also noted that two types of scenarios of particular importance include crossing trajectories and parallel trajectories. In agreement, we choose to develop scenarios of both types. We evaluated our methods on two separate experiments, one with detection ambiguity and one without.

Both experiments were implemented in the development software *julia* 0.4.3 [23] using the optimization package *JuMP* [24]. The implemented MIO utilized the optimization solver Gurobi 6.5.0[25]. Gurobi was limited to the use of a single

core for the optimization processes. Each simulation was run on a single node of the unclassified TX-Green cluster located at Lincoln Laboratories [26]. The cluster utilizes DL165 G7 compute nodes, consisting of 2.2 GHz compute nodes, with 8 GB of RAM each, for a total peak performance of 77.1 TFLOPS.

A. Experiment 1

In order to evaluate scalability we test our methods across a range of scenarios with varying numbers of targets and scans. In particular we consider: $P \in \{4, 6, 8, 10\}$ and $T \in \{4, 6, 8, 10\}$ seconds. Scans are collected at a rate of 1 Hz. The cartesian product of P and T creates 16 unique scenario sizes. We generate 10 unique crossing scenarios and 10 unique parallel scenarios of each size. Crossing scenarios have trajectories that intersect through time, while parallel scenarios have trajectories within close proximity to one another but do not ever actually intersect. Trajectories are restricted to exist within a fixed positional window of $[-10, 10]$. For each scenario, we randomly generate 10 realizations of data by perturbing each true position measurement by an error $\epsilon \sim \mathcal{N}(0, \sigma)$ with $\sigma \in \{0.1, 0.5, 1.0, 2.0, 3.5, 5.0\}$, where σ represents the noise parameter. The problem data is then generated by adding the detection error to the true position.

$$x_{it} = \alpha_i^{true} + \beta_i^{true} t + \epsilon \tag{37}$$

Scans $\mathcal{X}_t$ are simulated by randomizing the order of x_{it} for each t. Each unique $\boldsymbol{\mathcal{X}}$ generated is referred to as a *simulation*. For each such simulation, we run the heuristic with a range of starting points $N \in \{100 \quad 1,000 \quad 10,000\}$, and use each of these solutions as a warmstart for the MIO. The optimization process is set to terminate after 3T seconds, with solutions collected at intervals of $\{1, T, 2T, 3T\}$ seconds.

B. Experiment 2

The second experiment serves as an extension of the first in order to test the performance of our algorithms under detection ambiguity. We use the same base data generated from Experiment 1, but now we simulate missed detections and false alarms. A detection is removed with probability, γ, and we consider $\gamma \in \{0.8, 0.85, 0.9, 0.95\}$. For each scan, we generate false alarms by a poisson distribution with parameter, λ, which locations are then randomly selected uniformly within the state space. The false alarms are then added to $\mathcal{X}_t$ and the detection order of $\mathcal{X}_t$ is randomly shuffled. Once the data has been generated, we use the same approach as Experiment 1, testing our algorithms with identical values of N and capturing solutions at the same intervals. Additionally, we test the sensitivity of our methods across a range of penalties $\theta \in \{TBD\}$ and $\phi \in \{TBD\}$.

VIII. Computational Results

We begin by discussing the relationship between ρ and σ and discuss the how this relationship benefits both scenario generation and measuring complexity. Then we frame the performance of the basic heuristic before discussing the performance of the basic MIO model in both the data association and trajectory estimation spheres. We follow this with a discussion of the robust MIO model evaluated under both spheres.

A. Scenario Generation

Figure 1 below shows the relationship between σ and ρ. The plot is broken down by scenario type between crossing and parallel trajectories. It can be seen that according to ρ, the parallel method of scenario generation on the average creates easier scenarios for the data association problem. This trend would be expected because it reasons that crossing scenarios would be more likely to exhibit detections within close proximity. From this plot we also see how a small range of six values of σ corresponds to the full range of ρ from 0 to 1, meaning that we can quantify data association performance across a more continuous range.

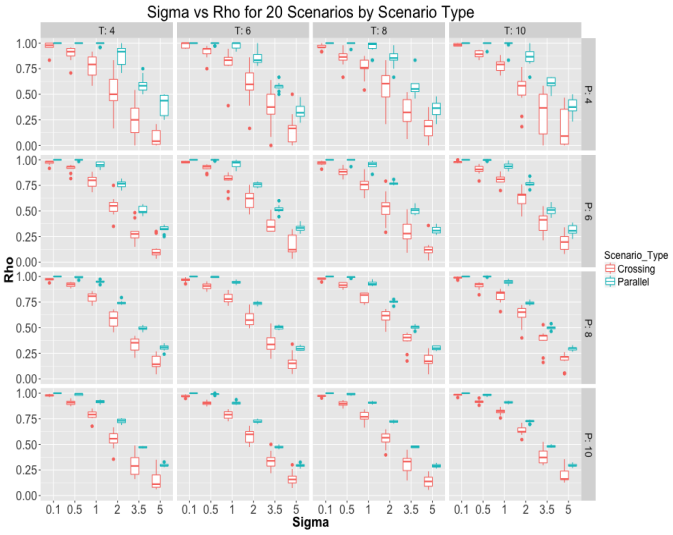

Fig. 1: Relationship between σ and ρ summarized by scenario type.

B. Basic Heuristic Results

We begin our discussion of the heuristic by evaluating the run times. Table I summarizes the average run times of the heuristic from Experiment 1 for each value of N, the number of heuristic starting points, arrange by the number of targets (P) and number of scans (T).

We see that run times increase linearly with the number of starting points for a fixed number of targets and scans. Because of this relationship, Table I could be used to easily compute the run time required for a given application. For a given scenario of targets and scans and a desired number of starting points, the required heuristic run time can be projected using this table. While this may seem to grow quickly for some scenarios, it is important to note that the power of the heuristic is that it can be leveraged through parallelization by running a subset of the total desired number of starting points on several processors. aThe linear relationship works to our advantage with parallelization because the total run time for all starting points can be divided by the number of processors to be used in parallelization to find the required run time for each

P	T	Number of starting points		
		100	1,000	10,000
4	4	0.01	0.10	0.86
4	6	0.02	0.24	2.18
4	8	0.05	0.45	4.28
4	10	0.08	0.76	7.34
6	4	0.01	0.15	1.41
6	6	0.04	0.39	3.67
6	8	0.09	0.81	7.87
6	10	0.17	1.56	15.34
8	4	0.02	0.19	1.83
8	6	0.07	0.57	5.49
8	8	0.14	1.24	12.19
8	10	0.27	2.53	25.15
10	4	0.03	0.25	2.38
10	6	0.09	0.80	7.87
10	8	0.20	1.84	18.48
10	10	0.39	3.73	37.44

TABLE I: Heuristic run times on a single processor (in seconds).

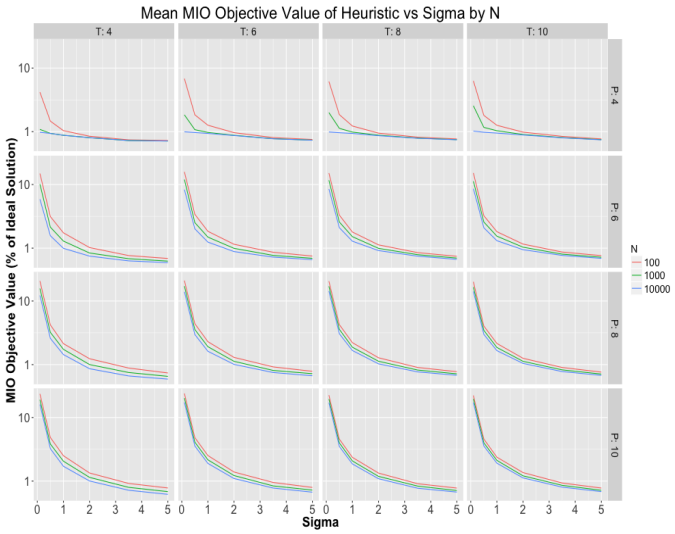

Fig. 2: Heuristic performance as a percentage of the ideal solution's MIO objective value.

subset of starting points. For example, consider a scenario of six targets over a period of six scans. If we desire to run the heuristic with 50,000 starting points, the total run time required when run in sequence is projected to be approximately 40 seconds according to table one. However, if we parallelize this onto 100 processors, then the real run time is reduced to 0.4 seconds. Thus, the run time of the heuristic can be reduced to meet efficiency needs subject only to the limitation of available processors.

We continue our evaluation of the basic heuristic by analyzing the performance of its solution on the MIO objective. This gives insight into wether or not the use of RSS as a proxy for the MIO objective is effective. To evaluate effectiveness on these terms, we compute the corresponding MIO objective value of the heuristic solution and normalize it against the MIO objective value of the *ideal* solution, which refers to the solution in which the data association problem is exactly correct (all detection assignments are exactly known). We refer to the resulting normalized value as *percentage of the ideal solution's MIO objective value*. Figure 2 plots this normalized against σ.

We see that increasing the number of starting points improves the quality of the heuristic solution as compared to the ideal solution's MIO objective value, especially when the number of targets is small. However, this effect is diminished as the number of targets increases. In addition, we see that for larger numbers of targets, even the largest number of starting points does not achieve near ideal performance, suggesting the need for a much larger number of starting points. This is not considered to be a problem, however, due to the advantages of parallelization discussed previously and also due to the power of optimization, which we will see later.

We also see that for larger values of σ the heuristic actually outperforms the ideal solution's MIO objective value. Remember that the ideal solution is simply ideal in the sphere of data association, while the MIO objective intends to score both the data association and trajectory estimation simultaneously. Therefore, we draw the conclusion that achieving perfect data association for large values of sigma does not necessarily correspond to the best solution to the trajectory estimation problem. In other words, as σ increases it may be necessary to tradeoff correct data associations in order to improve the trajectory estimation. We believe the results of the heuristic could be explained by this effect.

Next, we evaluate the performance of the basic heuristic on the data association problem as a function of the number of starting points. To this end, we relationship between accuracy and N. Figure 3 plots the mean accuracy of each of the three starting points from Experiment 1 against ρ.

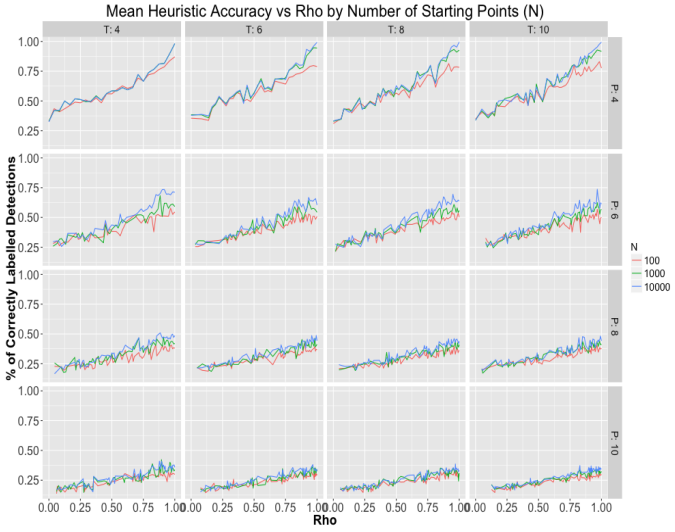

Fig. 3: Accuracy of basic heuristic by number of heuristic starting points.

First of all, we see that the heuristic finds good solutions to the data association problem, especially for scenarios with fewer targets, but performance degrades as the number of targets increases which is expected. Again it is seen that increasing the number of starting points results in minor improvements, and this improvement is greatest for scenarios with fewer targets. We see a similar effect as ρ increases.

We conclude that even small values of N produce moderately good solutions as measured by accuracy.

Overall we conclude that there is not a significant difference in heuristic performance for the range of N values that we explored. Therefore for simplification as we move forward in our analysis, we will restrict our discussions of the heuristic to $N = 1,000$.

C. Basic MIO Results

Next, we evaluate the accuracy of the MIO. Figure 4 plots the mean accuracy of the MIO, initialized by the $N = 1,000$ heuristic solutions, after 1,T, and 2T seconds against ρ. We have excluded the data for the MIO after 3T seconds for the sake of clarity as it showed little to no improvement over the MIO after 2T solution. For comparison, we have included the heuristic (for $N = 1,000$ only) in addition to a randomized solution, one in which we randomly assigned detections to targets. Note that in this case, that the ideal solution, one in which the associations are exactly correct, achieves an accuracy of 1.0 in all cases so it is not shown explicitly.

Fig. 4: Accuracy of MIO compared against the heuristic and a randomized solution.

For scenarios with fewer numbers of targets, the MIO solutions were actually proven to be the optimal solution. Therefore, for smaller scenarios with few targets, we see that the heuristic achieves optimal and near optimal solutions. We also see that the easier the scenario, the more improvement the MIO has over the heuristic, while in more difficult scenarios the effect is diminished. Furthermore, it can be seen that in nearly all scenarios, the MIO achieves its best or near best solutions after T or fewer seconds, suggesting the usefulness of the MIO as an online algorithm with a sliding window.

Next, we evaluate the performance of the basic heuristic and MIO through the lens of trajectory estimation. As discussed previously, we are interested in comparing δ, our proxy for ground track error, against σ, our measure of difficulty for trajectory estimation, in order to analyze performance of in the sphere of estimation. Figure 5 plots σ against δ for each of the solution types. In addition to the random solution shown on the previous plot, we also add a comparison to the ideal solution, as previously defined.

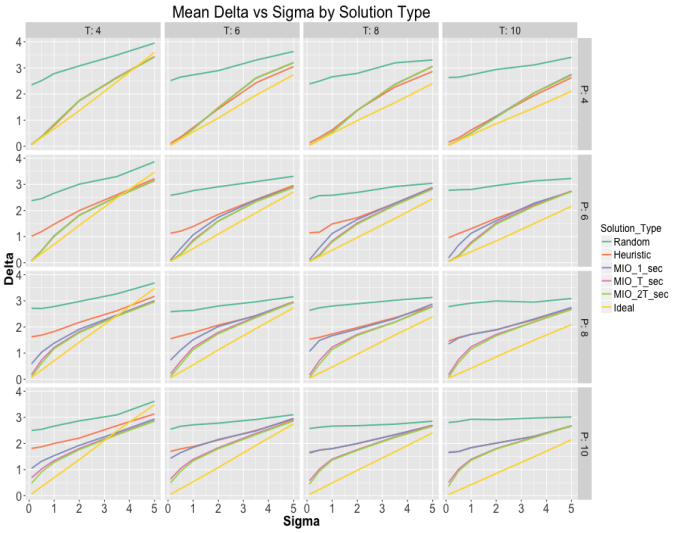

Fig. 5: Trajectory estimation performance

Remember that lower values of delta correspond to trajectory estimations that are closer to that of the true ground track. We see that the performance of the heuristic converges to that of the MIO for scenarios with few targets, as well as for large values of σ. Additionally, we see that as the number of targets increases we begin to see stronger improvements by the MIO over the heuristic. Interestingly, we see that for the scenarios with the largest number of targets and scans, the MIO after one second is not much better than the heuristic. While the MIO after T seconds provides significant improvement over that of the heuristic and MIO after 1 second, there is little further improvement in running the MIO for 2T seconds.

Again, we see that in scenarios with only for scans ($T = 4$) we see that for larger values of σ the heuristic and/or MIO sometimes outperforms the ideal. This is likely a result of limited data and increases uncertainty under high noise. As the number of scans approaches infinity, the ideal solution, or perfect data associations, leads to trajectory estimates that are closer and closer to the true ground track. Put differently. as more and more data is known, it becomes easier to estimate the trajectories even in the event of large noise, and so the trajectory estimates that result from the ideal solution converges to the true ground track.

D. Robust Heuristic & MIO Results

Results to be included in next draft.

IX. Conclusion and Future Work

We presented a multi-target tracking approach which jointly solves the problems of data association and trajectory estimation. We accomplish this without the need of a trajectory bank nor the a prior computation of trajectory hypothesis. We demonstrated that the proposed method outperforms for linear trajectories....

1) Online algorithm with sliding window 2) More complex penalties

ACKNOWLEDGMENTS

The authors would like to thank Sung-Hyun Son, Ph.D. and Steven Relyea at Lincoln Laboratories for their efforts in problem clarification as well as for providing feedback and other expert knowledge on the MTT problem and its intricacies. Additionally, we would like to thank Lincoln Laboratories and the LLGrid team for its assistance in running our experiments.

CONTRACT ACKNOWLEDGMENT: This material is based upon work supported by the Air Force under Air Force Contract No. FA8721-05-C-0002 and/or FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force.

REFERENCES

[1] G. Pulford, "Taxonomy of multiple target tracking methods," *Radar, Sonar and Navigation, IEE Proceedings* -, vol. 152, no. 5, pp. 291–304, October 2005.

[2] Y. Bar-Shalom and X. Li, *Multitarget-multisensor Tracking: Principles and Techniques*. Yaakov Bar-Shalom, 1995. [Online]. Available: https://books.google.com/books?id=GfOoMQEACAAJ

[3] Y. Bar-Shalom and X.-R. Li, *Estimation with Applications to Tracking and Navigation*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[4] S. Blackman, *Multiple-target Tracking with Radar Applications*, ser. Radar Library. Artech House, 1986. [Online]. Available: https://books.google.com/books?id=Ag9TAAAAMAAJ

[5] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[6] D. Reid, "An algorithm for tracking multiple targets," *Automatic Control, IEEE Transactions on*, vol. 24, no. 6, pp. 843–854, Dec 1979.

[7] S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 19, no. 1, pp. 5–18, Jan 2004.

[8] J. K. Wolf, A. M. Viterbi, and G. S. Dixon, "Finding the best set of k paths through a trellis with application to multitarget tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 25, no. 2, pp. 287–296, Mar 1989.

[9] D. Castanon, "Efficient algorithms for finding the k best paths through a trellis," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 2, pp. 405–410, Mar 1990.

[10] R. Perry, A. Vaddiraju, and K. Buckley, "Multitarget list viterbi tracking algorithm," in *Signals, Systems amp; Computers, 1998. Conference Record of the Thirty-Second Asilomar Conference on*, vol. 1, Nov 1998, pp. 436–440 vol.1.

[11] C. Morefield, "Application of 0-1 integer programming to multitarget tracking problems," *Automatic Control, IEEE Transactions on*, vol. 22, no. 3, pp. 302–312, Jun 1977.

[12] C. Carthel and S. Coraluppi, "Multi-hypothesis sonar tracking," in *Fusion 2004: Seventh International Conference on Information Fusion*, 2004.

[13] A. P. Poore and N. Rijavec, "A lagrangian relaxation algorithm for multidimensional assignment problems arising from multitarget tracking," *SIAM Journal on Optimization*, vol. 3, no. 3, pp. 544–563, 1993. [Online]. Available: http://dx.doi.org/10.1137/0803027

[14] A. Andriyenko and K. Schindler, "Multi-target tracking by continuous energy minimization," in *CVPR*, 2011.

[15] A. Andriyenko, K. Schindler, and S. Roth, "Discrete-continuous optimization for multi-target tracking," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 1926–1933.

[16] A. Lodi, *50 Years of Integer Programming 1958-2008.From the Early Years to the State-of-the-Art*, M. Jnger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, Eds. Springer Berlin Heidelberg, 2010.

[17] R. E. Bixby, "Mixed-integer programming: It works better than you may think," http://www.ferc.gov/CalendarFiles/20100609110044-Bixby,%20Gurobi%20Optimization.pdf, 2010, accessed 4 April 2016s.

[18] I. Gurobi Optimization, "Gurobi 6.5 performance benchmarks," http://www.gurobi.com/pdfs/benchmarks.pdf, 2015, accessed 4 April 2016s.

[19] G. Nemhauser, "Integer programming: Global impact," https://smartech.gatech.edu/bitstream/handle/1853/49829/presentation.pdf, 2013, accessed 4 April 2016s.

[20] Top500 Supercomputer Sites, "Performance development," http://www.top500.org/statistics/perfdevel/, 2015, accessed 4 April 2016s.

[21] B. Ristic, B. N. Vo, D. Clark, and B. T. Vo, "A metric for performance evaluation of multi-target tracking algorithms," *IEEE Transactions on Signal Processing*, vol. 59, no. 7, pp. 3452–3457, July 2011.

[22] C. Heij, P. de Boer, P. Franses, T. Kloek, H. van Dijk, and A. Rotterdam, *Econometric Methods with Applications in Business and Economics*. OUP Oxford, 2004. [Online]. Available: https://books.google.com/books?id=PJf6GUAavhAC

[23] N. J. Stor, I. Slapnicar, and J. L. Barlow, "Accurate eigenvalue decomposition of real symmetric arrowhead matrices and applications," *Linear Algebra and Its Applications*, vol. 464, pp. 62–89, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0024379513006265

[24] M. Lubin and I. Dunning, "Computing in operations research using julia," *INFORMS Journal on Computing*, vol. 27, no. 2, pp. 238–248, 2015. [Online]. Available: http://dx.doi.org/10.1287/ijoc.2014.0623

[25] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2015. [Online]. Available: http://www.gurobi.com

[26] N. Bliss, R. Bond, J. Kepner, H. Kim, and A. Reuther, "Interactive grid computing at lincoln laboratory," *MIT Lincoln Laboratory Journal*, vol. 16, no. 1, 2006.