Multi-target Tracking via Mixed Integer Optimization

by

Zachary Clayton Saunders

B.S. Operations Research, United States Air Force Academy (2014)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Master of Science in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2016

Author .....
Sloan School of Management
May 13, 2016

Certified by .....
Sung-Hyun Son
Assistant Group Leader, Lincoln Laboratory Group 36
Thesis Supervisor

Certified by .....
Dimitris Bertsimas
Boeing Professor of Operations Research
Co-Director, Operations Research Center
Thesis Supervisor

Accepted by .....
Patrick Jaillet
Dugald C. Jackson Professor
Department of Electrical Engineering and Computer Science
Co-Director, Operations Research Center

Multi-target Tracking via Mixed Integer Optimization

by

Zachary Clayton Saunders

Submitted to the Sloan School of Management
on May 13, 2016, in partial fulfillment of the
requirements for the degree of
Master of Science in Operations Research

Abstract

The field of multi-target tracking faces two primary challenges: (i) data association and (ii) trajectory estimation. Many algorithms such as the MHT and JPDAF attempt to solve the MTT problem via probabilistic approaches, while few employ the use of optimization models. In this paper we take a new approach to the MTT problem by introducing new mixed integer optimization (MIO) models that rely on minimal assumptions. These models solve the data association and trajectory estimation problems simultaneously by minimizing an easily interpretable global objective function. In addition, we propose a greedy heuristic that scales efficiently and finds high quality solutions in milliseconds, and we show that these heuristic solutions can be used effectively as a warm start for the MIO. We extend both the heuristic and the MIO model to scenarios with missed detections and false alarms. When there is no detection ambiguity *i.e.*, no false alarms or missed detections, our models require no penalties, and once we allow for detection ambiguity our models require the use of only two easily understood penalties. Furthermore, we introduce a new metric for measuring complexity in the data association problem as well as a metric for measuring the performance of trajectory estimation. Finally, we test all of our methods in two experimental simulations, one with detection ambiguity and one without, and we compare their performance with regards to randomized solutions and solutions with perfect data associations.

Thesis Supervisor: Sung-Hyun Son
Title: Assistant Group Leader, Lincoln Laboratory Group 36

Thesis Supervisor: Dimitris Bertsimas
Title: Boeing Professor of Operations Research
Co-Director, Operations Research Center

Acknowledgments

I would like to thank everyone who played a role in making this opportunity possible, everyone who supported me throughout this process, and everyone who influenced this project in any manner. Though I unfortunately do not have the space to thank each of you by name, I am immensely grateful for each and every one of you and the impacts you have had on me and this work.

I would like to thank my advisor, Professor Dimitris Bertsimas, for his ongoing support and guidance throughout this project. Without your direction and ideas none of this would have been possible. I thank you for constantly challenging me to push myself and for propelling me to expand my academic prowess beyond what I could have thought possible for myself. Additionally, I would like to thank Shimrit Shtern for her guidance and counseling on this project as well. Thank you for taking the time to pour over my error ridden scripts, edit this paper, and provide further guidance at critical points of this project.

Thank you to everyone at Lincoln Laboratories who played a role in making this degree possible. To Mr John Kuconis, thank you for facilitating this opportunity and generously sponsoring me through a military fellowship. To my advisor, Sung-Hyun Son, I also want to thank you for providing me the opportunity to be a Lincoln Laboratory Military Fellow in Group 36. Additionally, thank you for introducing me to the MTT problem and encouraging me to explore a field of the Air Force that was new to me, something that will no doubt pay dividends in my future as an officer in the Air Force. To Steve Relyea, thank you for meeting with me regularly and helping me weed through the details of the MTT problem, repeatedly scratching out ideas on paper. Your advice and insight was critical to both getting this project up and running and keeping it on track throughout the process. Furthermore, I would like to thank the Lincoln Laboratories LLGrid team for their support in running my experiments. As a first time Linux user, your assistance was instrumental in running my simulations and gathering my results.

Finally, I want to thank my friends and family. To my family, Mom, Dad, Tess,

Mathew, and Molly, thank you for your endless love and support and for constantly reminding me that hard work and due diligence always pays off. Without your continual support I would not be where I am today. I would also like to thank the truly extraordinary students at the MIT Operations Research Center who not only provided me with lasting friendships but also supported me academically both in the classroom and on this project.

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 and do not necessarily reflect the views of the U.S. Air Force.

Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

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 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 [21]. For a more exhaustive overview of estimation techniques, filtering, gating, and more please see [3] or [4].

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 [6]. Once the data association has been determined, the detections are often passed through one of numerous filters, most commonly a Kalman filter [14], 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 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 labeled 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 [22], 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 in [3].

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 [26] and an extension in [10] which formulates [26] as a network flow, reducing the solve time from exponential to polynomial. Still others, in particular [19], have suggested adaptations of this approach that output a single best set of K tracks, or a list of L best sets of k tracks, similarly to the MHT.

Compared to the number of statistically 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 (statis-

tically based) and hypotheses have been pruned [17]. Somewhat similarly, linear optimization has also been used to assist in the hypothesis selection process for the MHT [9]. Still, other attempts aim to improve the MHT hypothesis selection process via Lagrangian relaxation [20].

More recently, Andriyenko and Schindler have proposed formulating the MTT problem as a minimization of a continuous energy in [2] and then again as a minimization of discrete-continuous energy in [1]. 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 very efficiently [15]. 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 [5]. 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 [11]. CPLEX saw improvements by a factor of 29,000 from 1991 to 2007 [18]. From 1994 to 2014, the growth of supercomputing power as recorded by the TOP500 list has improved by a factor of 567839 [25]. 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 are negated. Recent work [23] 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 [23], but we show the value in relating a complexity measure to performance measures, namely that it allows us 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 the 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 provides 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 how to tune these parameters.

(iv) We present a new measure of complexity for the data association problem and show how it allows scenario generation and performance to be measured separately in each of their own natural demands. 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 Chapter 2. In Chapter 3 we introduce a simple MIO formulation for a sensor with no detection ambiguity and extend it to a generalized formulation. Then we develop a randomized local search heuristic in Chapter 4. In Chapter 5 we discuss extensions to both the MIO model and the heuristic to a sensor with detection ambiguity . Metrics for measuring scenario complexity and algorithm performance are discussed in Chapter 6. Experimental methods and computational results are presented in Chapter 7, including results for scenarios both with and without detection ambiguity. Finally, we summarize our contributions and identify future work in Chapter 8.

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

Chapter 2

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 in each scan, and there is no other source of detections. Therefore, the number of detections in each scan exactly equals the number of targets in existence. Under these conditions, the data association problem reduces to a one-to-one assignment problem. Our basic optimization model, presented in Chapter 3 aims to model this variant of the MTT problem.

Detection ambiguity refers to a more complex case where the sensor generates both false alarms and missed detections. A *false alarm* occurs when a detection is collected when in fact 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 in a given scan where a target does in fact actually exist. Now, the number of detections in each scan could be higher or lower than the actual number of existing targets. Under these conditions, each detection can now be assigned to

a target in the same manner as before, or the detection can be classified as a false alarm. Furthermore, we wish to identify the location (scan and target ID) of a missed detection. In Chapter 5 we make extensions of the formulation of our basic optimization model to a robust formulation that deals with this detection ambiguity, and we will refer to this formulation 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 in which T scans are collected. Scans occur at a fixed rate of 1Hz, such that the set of scans can be time stamped 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 paramaterized by a true initial position α_j^{true} and a true constant velocity β_j^{true}.

Chapter 3

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 scan i.e., no missed detections.

(ii) The sensor does not generate any additional detections i.e., no false alarms.

We begin constructing our MIO model by introducing decision variables that define data associations as well as estimated trajectories. Using these decision variables, we then develop an objective function that mathematically quantifies the value of the model decisions. Finally, we restrict these variables using constraints that force the model to find solutions that are feasible for the MTT problem as we have defined it. A simple model is first developed step by step in the coming sections before generalized formulation is presented.

3.0.1 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, respectively, 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 association 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:

$$y_{itj} = \begin{cases} 1, & \text{if detection } x_{it} \text{ is assigned to trajectory } j, \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

Next, we use these decision variables to develop an objective function to score the solutions found by the model.

3.0.2 Objective Function

Here, we would like to develop a function that quantifies the quality of a feasible solution. Our goal is to produce a single measure for both the data association and the trajectory estimation problems. For a given assignment and a given estimated trajectories we define the quality of the estimation as the distance of each detection from the estimated position of its associated trajectory. That is, if at scan t detection x_{it} is associated with trajectory j then, the distance

$$\|x_{it} - \hat{x}_{jt}\|,$$

is the measure of the quality of estimation for trajectory j at scan t, and the total estimation quality for a given association will be given as

$$\sum_{j=1}^{P} \sum_{t=1}^{T} \left\| \sum_{(i,j)\in\mathcal{A}_t} x_{it} - \hat{x}_{jt} \right\|, \tag{3.2}$$

where $\mathcal{A}_t$ is pairs of detection-trajectory associations made for scan t.

We can now separate the problem into two parts: given an assignment finding

the estimated trajectories which minimizes (3.2), and finding the assignment which results in the best estimated trajectories. Recall that each trajectory is defined by two parameters, α_j^{true} and $\beta_j^{\text{true}} t$ such that the true location is given as

$$\bar{x}_{jt} = \alpha_j^{\text{true}} + \beta_j^{\text{true}} t. \tag{3.3}$$

Thus, an estimated trajectory can analogically be defined by α_j and β_j such that its estimated location at the time of scan t is given by

$$\hat{x}_{jt} = \alpha_j + \beta_j t. \tag{3.4}$$

Therefore, given a full assignment $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_T)$, the trajectory which has the best estimation error is actually the one which solves the problem

$$\underset{\alpha_j, \beta_j}{\text{minimize:}} \sum_{j=1}^{P} \sum_{t=1}^{T} \left\| \sum_{(i,j) \in \mathcal{A}_t} x_{it} - (\alpha_j + \beta_j t) \right\|, \tag{3.5}$$

Notice that under the current assumptions, in which there is no detection ambiguity, (3.5) is the cost of assignment $\mathcal{A}$.

Now we turn to the problem of choosing the assignment, based on this measure. To this end we formulate the assignment cost (3.5) in terms of our decision variables. Note that $(i, j) \in \mathcal{A}_t$ if and only if $y_{itj} = 1$, and because all detections will be assigned to a trajectory and vice versa, the following equivalence holds:

$$\sum_{(i,j) \in \mathcal{A}_t} x_{it} = \sum_{i=1}^{P} y_{itj} x_{it}. \tag{3.6}$$

Making the appropriate substitutions, the cost of an assignment described by variables y_{itj} is given as:

$$\sum_{j=1}^{P} \sum_{t=1}^{T} \left\| \sum_{i=1}^{P} y_{itj} x_{it} - (\alpha_j + \beta_j t) \right\|. \tag{3.7}$$

Therefore, in order to find the assignment with the lowest cost, we are left to minimize cost (3.7) over all assignments, and obtain the following final objective:

$$\underset{y_{itj},\alpha_j,\beta_j}{\text{minimize:}} \sum_{j=1}^{P}\sum_{t=1}^{T}\left\|\sum_{i=1}^{P} y_{itj}x_{it} - (\alpha_j + \beta_j t)\right\|. \tag{3.8}$$

At this point it is necessary to discuss the advantages and disadvantages of the two natural distance measures (norms) that will be considered: the ℓ_1 and the ℓ_2 norms. The ℓ_1 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. Furthermore, existing algorithms for MIO are more well developed for linear rather than quadratic optimization. However, the ℓ_2 norm squared form, which is equivalent to the residual sum of squares (RSS), has the advantage that it can be quickly computed using simple linear algebra, making it more amenable to a heuristic. This concept will be discussed further in Chapter 4.

Because of the computational benefits of linear optimization over quadratic optimization, we choose to formulate the objective using the ℓ_1 norm. Therefore, we now show how (3.8) can be reformulated using linear optimization in the case of the ℓ_1 norm by introducing continuous variables $\psi_{jt} \in \mathbb{R}^n$ and the following constraints:

$$\sum_{i=1}^{P} y_{itj}x_{it} - \alpha_j - \beta_j t \leq \psi_{jt}, \qquad \forall j, t, \tag{3.9}$$

$$-\left(\sum_{i=1}^{P} y_{itj}x_{it} - \alpha_j - \beta_j t\right), \geq \psi_{jt} \qquad \forall j, t. \tag{3.10}$$

The resulting objective function for the case of the ℓ_1 norm would then be:

$$\underset{\psi_{jt}}{\text{minimize:}} \sum_{j=1}^{P}\sum_{t=1}^{T}\psi_{jt}. \tag{3.11}$$

3.0.3 Constraints

In addition to the constraints used to linearize the objective function, we also require standard assignment constraints to ensure that only one detection is assigned

to a target and vice versa. Specifically, for each scan, each detection x_{it} must be assigned to exactly one target j:

$$\sum_{j=1}^{P} y_{itj} = 1 \qquad \forall i, t. \tag{3.12}$$

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

$$\sum_{i=1}^{P} y_{itj} = 1 \qquad \forall j, t. \tag{3.13}$$

3.0.4 Simple Formulation

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

$$\begin{aligned}
\underset{\psi_{jt}}{\text{minimize:}} \quad & \sum_{j=1}^{P}\sum_{t=1}^{T} \psi_{jt} \\
\text{subject to:} \quad & \sum_{j=1}^{P} y_{itj} = 1 \qquad \forall i, t \\
& \sum_{i=1}^{P} y_{itj} = 1 \qquad \forall j, t \\
& \sum_{i=1}^{P} y_{itj} x_{it} - \alpha_j - \beta_j t \leq \psi_{jt} \qquad \forall j, t \\
& -\left(\sum_{i=1}^{P} y_{itj} x_{it} - \alpha_j - \beta_j t\right) \geq \psi_{jt} \qquad \forall j, t \\
& y_{itj} \in \{0, 1\} \quad \forall i, t, j \\
& \alpha_j \in \mathbb{R}^n \quad \forall j, \quad \beta_j \in \mathbb{R}^n \quad \forall j, \quad z_{jt} \in \mathbb{R}^n \quad \forall j, t
\end{aligned} \tag{3.14}$$

This formulation is simple in the sense that it involves few variables and constraints, making it highly interpretable and easily implementable. However, it has the disadvantage of being ill suited for extensions to detection ambiguity because it heavily relies on the fact that exactly one of the detections at each scan is associated to a target. This forces the term $\sum_{i=1}^{P} y_{itj} x_{it}$ to always be equal to one of the detections.

However, in the case of detection ambiguity, this no longer holds true, since there might be trajectories which are not associated with a detection in a given scan, resulting in an unintended cost to the assignment. Therefore, in the following section we present a more generalized formulation, which is amenable to scenarios with detection ambiguity.

3.0.5 Generalized Formulation

Here we modify formulation (3.14) so that it can be easily extended to handle false alarms and missed detections that occur in scenarios with detection ambiguity. We previously identified that the main issue with (3.14) arises from the fact that $\sum_{i=1}^{P} y_{itj} x_{it}$ will always incur a cost in the objective function. However, we only wish to account for this assignment cost when a target has actually been assigned. To this end, we introduce a new variable z_{jt} to substitute for this term. We will then force this variable to take the value x_{it} when $\sum_{j=1}^{P} y_{itj} = 1$ and some arbitrary number when $\sum_{j=1}^{P} y_{itj} y_{itj} = 0$.

$$z_{jt} = \begin{cases} x_{it}, & \text{if } \sum_{j=1}^{P} y_{itj} = 1, \\ \textit{free}, & \text{otherwise.} \end{cases} \tag{3.15}$$

This logical condition can be translated to a model constraint through the following constraint:

$$M_t(1 - y_{itj}) \geq |z_{jt} - x_{it} y_{itj}| \qquad \forall i, t, j. \tag{3.16}$$

where $M_t = \max_j |x_{it}|$ for each scan. Furthermore, we can linearize this constraint by substituting it for the following two linear constraints:

$$x_{it} y_{itj} + M_t(1 - y_{itj}) \geq z_{jt} \qquad \forall i, t, j, \tag{3.17}$$

$$x_{it} y_{itj} - M_t(1 - y_{itj}) \leq z_{jt} \qquad \forall i, t, j. \tag{3.18}$$

We must also adjust (3.8) to account for this change as follows:

$$\underset{z_{jt},\alpha_j,\beta_j}{\text{minimize:}} \sum_{j=1}^{P}\sum_{t=1}^{T} \|z_{jt} - \alpha_j - \beta_j t\|. \tag{3.19}$$

This objective can be linearized in the same fashion as before by again introducing continuous variables ψ_{jt} and additional constraints as follows:

$$\underset{\psi_{jt}}{\text{minimize:}} \sum_{j=1}^{P}\sum_{t=1}^{T} \psi_{jt} \tag{3.20}$$

$$z_{jt} - \alpha_j - \beta_j t \leq \psi_{jt} \qquad \forall i, j, t, \tag{3.21}$$

$$-(z_{jt} - \alpha_j - \beta_j t) \geq \psi_{jt} \qquad \forall i, j, t. \tag{3.22}$$

Again, we consolidate these elements together and arrive at the following generalized MIO model:

$$\begin{aligned}
\underset{\psi_{jt}}{\text{minimize:}} \quad & \sum_{j=1}^{P}\sum_{t=1}^{T} \psi_{jt} && && (3.23)\\
\text{subject to:} \quad & \sum_{j=1}^{P} y_{itj} = 1 && \forall i, t\\
& \sum_{i=1}^{P} y_{itj} = 1 && \forall j, t\\
& x_{it} y_{itj} + M_t(1 - y_{itj}) \geq z_{jt} && \forall i, t, j\\
& x_{it} y_{itj} - M_t(1 - y_{itj}) \leq z_{jt} && \forall i, t, j\\
& z_{jt} - \alpha_j - \beta_j t \leq \psi_{jt} && \forall i, j, t\\
& -(z_{jt} - \alpha_j - \beta_j t) \geq \psi_{jt} && \forall i, j, t\\
& y_{itj} \in \{0, 1\} \quad \forall i, t, j\\
& \alpha_j \in \mathbb{R}^n \quad \forall j, \quad \beta_j \in \mathbb{R}^n \quad \forall j, \quad z_{jt} \in \mathbb{R}^n \quad \forall j, t
\end{aligned}$$

Note that (3.14) and (3.23) are exactly identical formulations when detection

ambiguity does not exist. In Chapter 5 we will extend (3.23) to account for false alarms and missed detections under sensor ambiguity.

Chapter 4

Heuristic

In this section, we present a detailed description of a heuristic that finds high quality feasible solutions for problem (**??**). 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 randomized local search methods to find locally optimal solutions.

The fundamental concept behind randomized local search methods is to begin with a randomized starting point and through local improvements converge to a locally optimal solution. By applying this scheme to a growing number of randomized starting points, the probability of reaching high quality solutions, or even the globally optimal solution, increases.

We now detail the heuristic mechanism for a single starting point. The heuristic *starting point* is a randomized solution to the assignment problem, that satisfies the assignment equations (3.12) and (3.13). This is done for each scan, by choosing a permutation over the detections uniformly at random, and associating to trajectories by the order of the permutation, i.e., if in the random permutation detection i is at location j than it will be assigned to trajectory j. The assignment cost, temporarily denoted by F, of this starting point is then calculated by solving (3.7). After initializing the heuristic starts a *sweep* through the scans, i.e., a single pass through the scans. At each stage of the sweep two detections are randomly selected from the current scan and they exchange assignments, an operation we call a *swap*, generating a new feasible solution. The assignment cost of this new solution is calculated, and if

it improves the current cost it is kept and otherwise it is rejected. The heuristic will continue to conduct sweeps, until a full sweep is completed without accepting a single swap, in which point it terminates. The full description of the heuristic algorithm for a single starting point is given in Figure 4-1.

Algorithm 1 Randomized local search with heuristic swaps

Input: $\boldsymbol{\mathcal{X}}$, P, T
Output: F, y_{itj}
Initialization : Assign random initial assignments for y^0_{itj}
Calculate $\alpha_j, \beta_j \quad \forall j$
Calculate F^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 $F^k, \alpha_j, \beta_j, \alpha_m, \beta_m$
    if $(F^k \geq F^{k-1})$ **then**
      $y^k \leftarrow y^{k-1}$
    else
      swapped $\leftarrow true$
    end if
  end for
  $k \leftarrow k + 1$
end while
return F^k, y^k_{itj}

Figure 4-1: Pseudocode for heuristic for a single starting point.

The goal of any heuristic is to find good feasible solutions in an efficient manner. In particular, the goal of our heuristic is to find good feasible solution for the MIO formulation, which can serve as a warm start for the MIO. In Chapter 3 we discussed our choice to use the ℓ_1 norm over the ℓ_2 norm for use in the objective of our MIO models. We now turn to discuss why the ℓ_2 is the preferred choice for use in this heuristic. The two main ares of concern are 1) efficiency of the algorithm and 2)

quality of the solution.

In the case of the MIO, the preferred objective function was the ℓ_1 norm because it lent itself easily to linear optimization solvers which have known performance advantages over quadratic optimization solvers. However, in the case of the heuristic, the objective function is computed given an assignment, i.e., we need to solve problem (3.7), to obtain the specific assignment's cost, rather than the more difficult problem (3.8). Furthermore, this objective needs to be recomputed after each swap, even if it is eventually rejected, and hundreds to thousands of swaps may be carried out for a single starting point. This fact makes the computational cost of this calculation critical to the scalability of the heuristic. Computing the ℓ_1 norm objective would require solving an linear optimization problem. Even though this linear optimization problem can be computed quite quickly by state of the art optimization solvers, the ℓ_2 norm squared, or RSS, can be computed by simple linear algebra in a fraction of the time, as shown in [13]. Therefore, with respect to efficiency, the ℓ_2 norm is the clear choice for use in the heuristic.

When judging the quality of the heuristic solution, we must look at its purpose. Since it will serve as a warm start for the MIO, which uses the ℓ_1 norm in its objective, we would assume that better solutions will be obtained by using the same norm for the heuristic objective. However, the heuristic will converge to high quality solutions, and since both norms represent measures of distance, and hence are correlated, we assume that a high quality solution as measured by the ℓ_2 norm is also a high quality solution as measured by the ℓ_1 norm. Thus, the choice of the ℓ_2 over the ℓ_1 norm might not significantly degrade the solution quality.

Although the ℓ_2 norm runs the risk of searching for lower quality solutions, due to its susceptibility to outliers, the potential loss in solution quality is far outweighed by the guaranteed efficiency improvements afforded by the ℓ_1 norm. Furthermore, this risk can be mitigated through the use of parallelization. To elaborate, the heuristic described above can be parallelized by running partitions of the starting points on separate cores, meaning that a large number of starting points can run in a relatively short amount of time. Increasing the number of starting points greatly reduces the

potential for the heuristic to get stuck at a poor quality solution. Therefore, we make the choice to use the ℓ_2 norm in the objective function of the heuristic.

Chapter 5

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, we replace Assumption 2 with the following less restrictive assumption.

Assumption 3.

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.

5.0.6 Fixed Number of Targets

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 run in parallel.

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.

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

Constraints

All detections must either be assigned to a trajectory j or to a false alarm.

$$\sum_{j=1}^{P} y_{itj} + F_{it} = 1 \qquad \forall i, t \tag{5.1}$$

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

$$\sum_{i=1}^{n_t} y_{itj} + M_{jt} = 1 \qquad \forall j, t \tag{5.2}$$

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

$$\sum_{i=1}^{n_t} \sum_{t=1}^{T} F_{it} = TF \tag{5.3}$$

$$\sum_{j=1}^{P} \sum_{t=1}^{T} M_{jt} = TM \tag{5.4}$$

Objective Function

We can easily extend (3.19) to account for false alarms and missed detections by introducing penalties θ (ϕ, respectively) for each missed detection (false alarm, respectively). An objective function of this type would take the form of:

$$\underset{\psi_{jt}}{\text{minimize:}} \ \sum_{j=1}^{P} \sum_{t=1}^{T} \psi_{jt} + \theta \cdot TF + \phi \cdot TM. \tag{5.5}$$

which can be linearized in the same manner as (3.20). A discussion on the insight behind these penalties, and recommendations for tuning them can be found in Appendix B.

Formulation 2

$$
\begin{aligned}
\underset{\psi_{jt}}{\text{minimize:}} \quad & \sum_{j=1}^{P}\sum_{t=1}^{T} \psi_{jt} + \theta \cdot TF + \phi \cdot TM && (5.6)\\
\text{subject to:} \quad & \sum_{j=1}^{P} y_{itj} + F_{it} = 1 \qquad \forall i,t\\
& \sum_{i=1}^{n_t} y_{itj} + M_{jt} = 1 \qquad \forall j,t\\
& \sum_{i=1}^{n_t}\sum_{t=1}^{T} F_{it} = TF\\
& \sum_{j=1}^{P}\sum_{t=1}^{T} M_{jt} = TM\\
& x_{it}y_{itj} + M_t(1-y_{itj}) \geq z_{jt} \qquad \forall i,t,j\\
& x_{it}y_{itj} - M_t(1-y_{itj}) \leq z_{jt} \qquad \forall i,t,j\\
& z_{jt} - \alpha_j - \beta_j t \leq \psi_{jt} \qquad \forall j,t\\
& -(z_{jt} - \alpha_j - \beta_j t) \leq \psi_{jt} \qquad \forall j,t\\
& y_{itj} \in \{0,1\} \quad \forall i,t,j\\
& \alpha_j \in \mathbb{R}^n, \quad \beta_j \in \mathbb{R}^n \quad \forall j\\
& z_{jt} \in \mathbb{R}^n, \quad \forall j,t.
\end{aligned}
$$

5.0.7 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.

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.

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

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 to a false alarm. This can be implemented similarly to (5.1), except now we sum over N_1 because the number of targets is unknown but limited by N_1.

$$\sum_{j=1}^{N_1} y_{itj} + F_{it} = 1 \qquad \forall i, t. \tag{5.7}$$

Similarly, (5.4) must be adjusted to sum over the maximal number of targets allowed N_1.

$$\sum_{j=1}^{N_1} \sum_{t=1}^{T} M_{jt} = TM. \tag{5.8}$$

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

$$\sum_{i=1}^{n_t} y_{itj} + M_{jt} = w_j \qquad \forall j, t. \tag{5.9}$$

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.

$$|\alpha_j| + |\beta_j| \leq M_0 w_j \qquad \forall j. \tag{5.10}$$

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

$$w_{N_0+1} \geq ... \geq w_{N_1}, \tag{5.11}$$

which reduces the symmetry of the formulation, and thus making it efficiently solvable. Furthermore, we can replace (5.9) with the following two constraints:

$$\sum_{i=1}^{n_t} y_{itj} + M_{jt} = 1 \qquad \forall j = 1, ..., N_0, t \tag{5.12}$$

$$\sum_{i=1}^{n_t} y_{itj} + M_{jt} = w_j \qquad \forall j = N_0, ..., N_1, t. \tag{5.13}$$

Formulation 3

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

$$
\begin{aligned}
\underset{\psi_{jt}}{\text{minimize:}} \quad & \sum_{j=1}^{N_1}\sum_{t=1}^{T}\psi_{jt} + \theta\cdot TF + \phi\cdot TM && (5.14)\\
\text{subject to:} \quad & \sum_{j=1}^{N_1} y_{itj} + F_{it} = 1 \qquad \forall i,t\\
& \sum_{i=1}^{n_t} y_{itj} + M_{jt} = 1 \qquad \forall j = 1,...,N_0,t\\
& \sum_{i=1}^{n_t} y_{itj} + M_{jt} = w_j \qquad \forall j = N_0,...,N_1,t\\
& \sum_{i=1}^{n_t}\sum_{t=1}^{T} F_{it} = TF\\
& \sum_{j=1}^{N_1}\sum_{t=1}^{T} M_{jt} = TM\\
& w_{N_0+1} \geq ... \geq w_{N_1}\\
& |\alpha_j| + |\beta_j| \leq M_0 w_j \qquad \forall j\\
& x_{it}y_{itj} + M_t(1-y_{itj}) \geq z_{jt} \qquad \forall i,t,j\\
& x_{it}y_{itj} - M_t(1-y_{itj}) \leq z_{jt} \qquad \forall i,t,j\\
& z_{jt} - \alpha_j - \beta_j t \leq \psi_{jt} \qquad \forall j,t\\
& -(z_{jt} - \alpha_j - \beta_j t) \leq \psi_{jt} \qquad \forall j,t\\
& y_{itj} \in \{0,1\} \quad \forall i,t,j\\
& \alpha_j \in \mathbb{R}^n, \quad \beta_j \in \mathbb{R}^n, \quad w_j \in \mathbb{R}^n \quad \forall j\\
& z_{jt} \in \mathbb{R}^n, \quad \forall j,t.
\end{aligned}
$$

5.0.8 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.

Chapter 6

Scenario Complexity & Performance Metrics

There does not exist a unified approach for measuring scenario complexity as stated by [21] 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.

$$Accuracy = \frac{\#\text{ correct assignments}}{\text{Total }\#\text{ of detections}} = \frac{\#\text{ correct assignments}}{PT} \tag{6.1}$$

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

$$Accuracy = \frac{\#\text{ correct assignments}}{PT + \#\text{ False Alarms}}. \tag{6.2}$$

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.

$$D_{ijt} = \|\alpha_i^{\text{true}} + \beta_i^{\text{true}} t - \alpha_j^{\text{true}} + \beta_j^{\text{true}} t\|. \tag{6.3}$$

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.

$$\rho = \frac{\sum_{t=1}^{T} \sum_{i<j} c_{ijt}}{\binom{P}{2} T}. \tag{6.4}$$

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.

$$\delta = \frac{\sum_{t=1}^{T} \sum_{j=1}^{P} \|\bar{x}_{jt} - \hat{x}_{jt}\|}{PT}. \tag{6.5}$$

We match the true trajectories to the estimated trajectories using an one-to-one assignment problem which can be formulated using linear optimization. See Appendix A for more details. Lower values of δ correspond to higher performance

because the distance between the estimated and true ground trajectories is smaller.

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

Chapter 7

Experimental Simulations & Computational Results

There does not exist among the literature a clearly defined comprehensive set of standard test scenarios as pointed out by [21], which also notes that two types of scenarios of particular importance include crossing trajectories and parallel trajectories. With this in mind, we choose to generate scenarios of both trajectory types using a simple methodology that will be outlined in the following section. Furthermore, we evaluate our algorithms on two separate experiments, one with detection ambiguity and one without, with the first first outlined here in this section and the second outlined in the next.

Both experiments and all steps including the scenario generation process, heuristic, and MIO were implemented in the development software *julia* 0.4.3 [24] using the optimization package *JuMP* [16]. The optimization software Gurobi 6.5.0 [12] was used to solve the MIOs, and the optimization processes was restricted to the use of a single core. Each simulation was run on a single compute node of the unclassified TX-Green cluster located at Lincoln Laboratories. The cluster utilizes DL165 G7 compute nodes, consisting of 2.2 GHz compute cores, with 8 GB of RAM each, for a total peak performance of 77.1 TFLOPS [8].

7.0.9 Basic Scenario

In order to evaluate scalability of our algorithms 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. To generate trajectories, a grid size is first selected and then starting and stopping points are randomly selected within this grid. For crossing trajectories, the starting and stopping points have no restrictions, while for parallel trajectories the starting and stopping points are restricted to fall within subsets of the grid and these subsets are not allowed to overlap, resulting in trajectories that fall within close proximity to one another but do not overlap. For our experiments, a grid size of $[-10, 10]$ was used. For each scenario, we randomly generate 10 realizations of data by first 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:

$$x_{it} = \alpha_i^{\text{true}} + \beta_i^{\text{true}} t + \epsilon. \tag{7.1}$$

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.

We begin the analysis of the methodologies by discussing the relationship between ρ and σ and discuss how this relationship benefits both scenario generation and complexity measuring by allowing each to occur in their own natural domain. 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.

Scenario Generation

Figure 7-1 shows the relationship between σ and ρ for our scenario generation methods. The plot is broken down by scenario type between crossing and parallel trajectories.

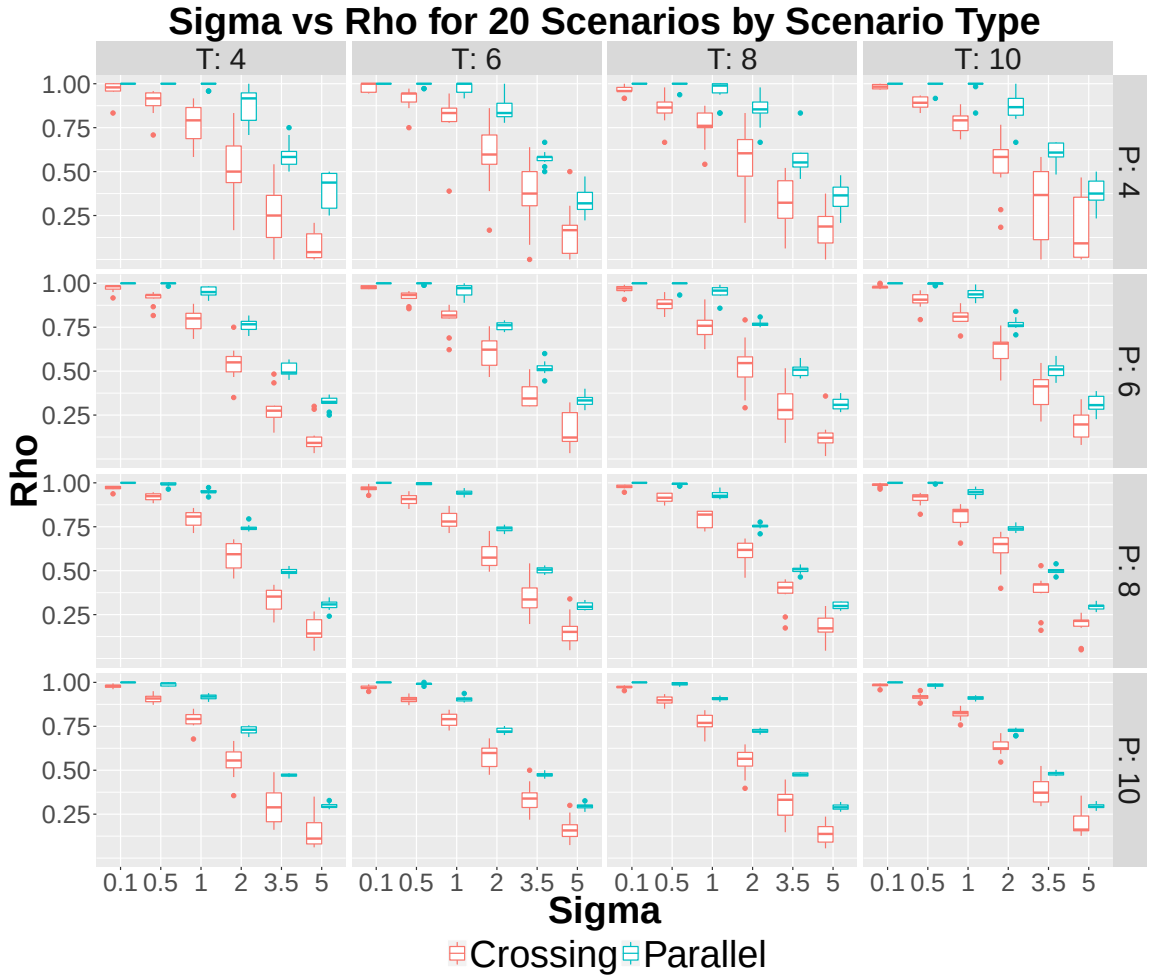

Figure 7-1: Relationship between σ and ρ summarized by scenario type for all 20 generated scenarios in this experiment.

It can be seen that the parallel method of scenario generation on the average creates easier scenarios for the data association problem, as measured by ρ,. This result supports the hypothesis that crossing scenarios would be more likely to exhibit detections within close proximity, and consequently result in more difficult scenarios as it pertains to the data association problem.

In addition, we see from this plot that a relatively 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. This means that σ can be used in its natural domain of the data generation process, and ρ can be back calculated as a measurement of difficulty for the data association problem. As a result, we gain a highly interpretable performance metric for the data association problem without sacrificing the ability to generate scenarios in their natural domain.

Basic Heuristic

We begin our discussion of the heuristic with an examination of the run times from the experiment. Table 7.1 summarizes the minimum, mean, and maximum run times of the heuristic from this experiment for a single starting point, arranged by the number of targets (P) and number of scans (T). Times are shown in milliseconds.

		Heuristic Run Times (in milliseconds)		
P	T	Min	Mean	Max
4	4	0.07	0.10	0.18
4	6	0.18	0.24	0.38
4	8	0.34	0.45	0.62
4	10	0.58	0.76	1.02
6	4	0.11	0.15	0.25
6	6	0.31	0.39	0.58
6	8	0.64	0.81	1.05
6	10	1.24	1.56	2.02
8	4	0.14	0.19	0.30
8	6	0.46	0.57	0.86
8	8	0.95	1.24	1.58
8	10	2.07	2.53	3.37
10	4	0.19	0.25	0.41
10	6	0.63	0.80	1.03
10	8	1.44	1.84	2.44
10	10	2.96	3.73	4.56

Table 7.1: Heuristic run times (in milliseconds) for a single starting point.

Notice that the number of scans has a much greater effect on run times than the number of targets. Whereas the average run time roughly doubles with each increase of 2 scans, the average run time increases as little as 27% but no more than 50% for

each increase of 2 targets. Therefore, the heuristic scales much more efficiently with P than T. Although this may initially seem like a cause for concern, this is actually a desirable trait because it suggests that the heuristic is likely a good candidate for use in a sliding window algorithm. Rather than solve all scans in a single batch at once, a sliding window algorithm solves a subset of scans, or a smaller window, and advances through all scans sequentially. As the window progresses forward through the scans, "soft" decisions are made meaning that the heuristic would begin with the decisions from the previous solution. As scans pass beyond the horizon and out of the sliding window, the decisions become fixed and we refer to them as "hard" decisions. This process continues until all scans have been processed. The run times of a sliding window variant of the heuristic would not exhibit the curse of dimensionality in T since the number of scans remains constant. Additionally, the heuristic is likely to produce higher quality solutions as a result of these "soft" decisions of previous steps since it is starting from a solution which is likely to be better than a completely random solution.

Furthermore, the heuristic is extremely parallelizable. This means that we can run different sets of starting points for the heuristic on several computers, allowing either a reduction in total run time for a fixed number of starting points or an increase in the number of starting points for a fixed run time. For example, the average run time for a single starting point of a scenario of six targets and six scans is about 0.4 milliseconds. If we desire to evaluate 50,000 heuristic starting points, we would require about 20 seconds of run time if run in sequence. However, the same number of starting points parallelized onto 100 processors would reduce the total run time to approximately 0.2 seconds. Thus, the run time of the heuristic can be reduced to meet the efficiency needs of the system 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 whether 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). The resulting normalized value represents the MIO objective value as a factor of the ideal solution's MIO objective value. This metric is plotted against σ and summarized in Figure 7-2.

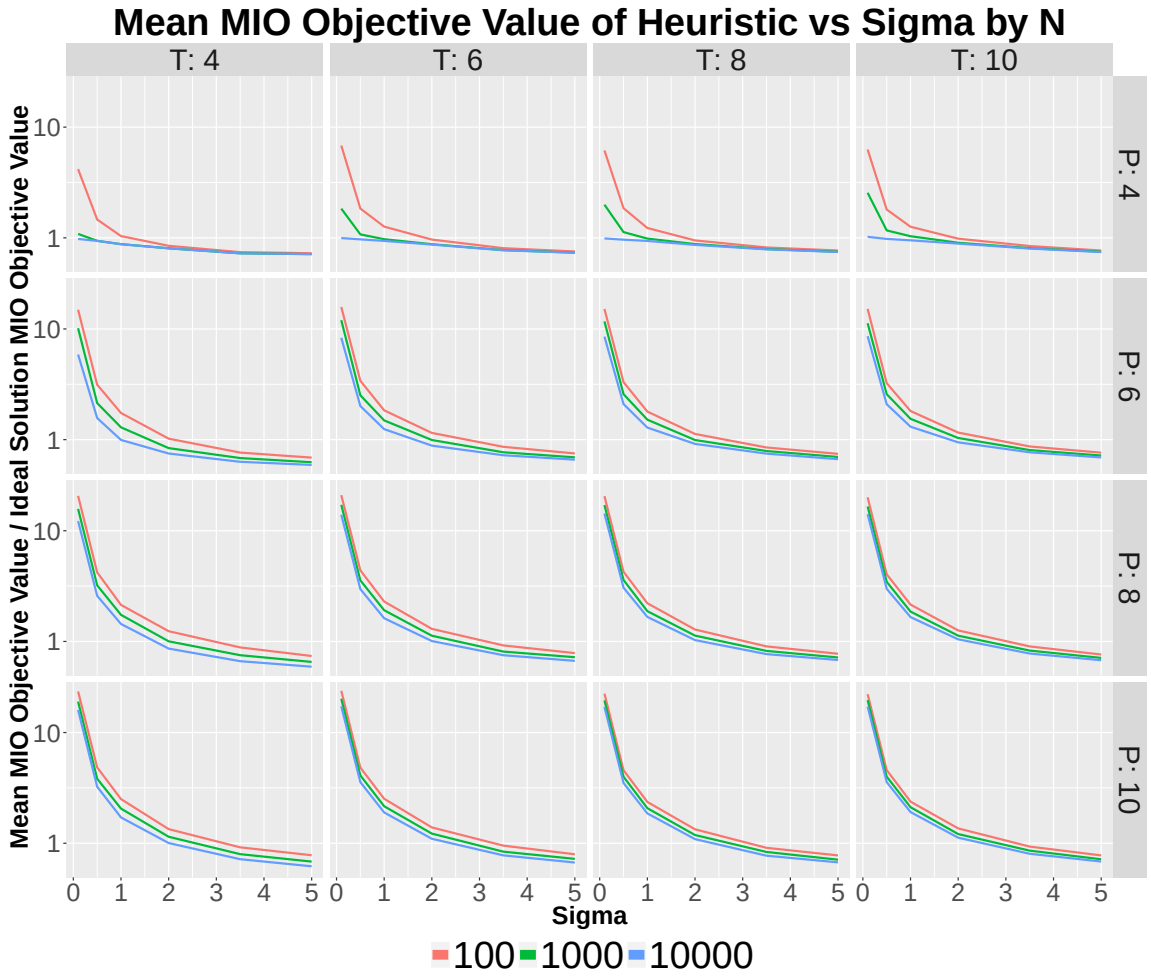

Figure 7-2: Heuristic performance as a factor of the ideal solution's MIO objective value.

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 7-3 plots the mean accuracy of each of the three starting points from this experiment against ρ.

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 $N = 1,000$.

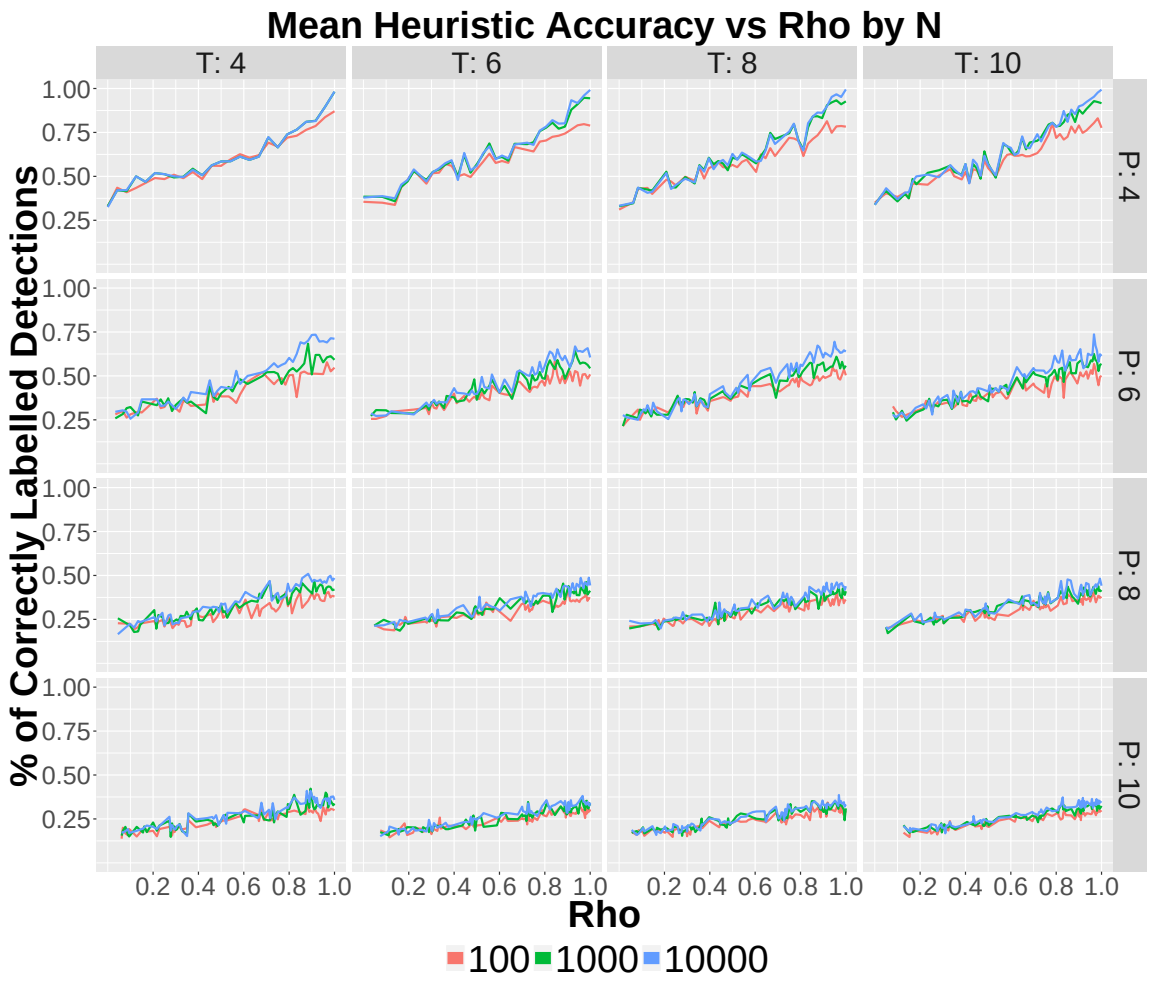

Figure 7-3: Accuracy of basic heuristic by number of heuristic starting points.

Basic MIO

Next, we transition our evaluation to focus on the MIO by first measuring its performance on the data association problem. Figure 7-4 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.

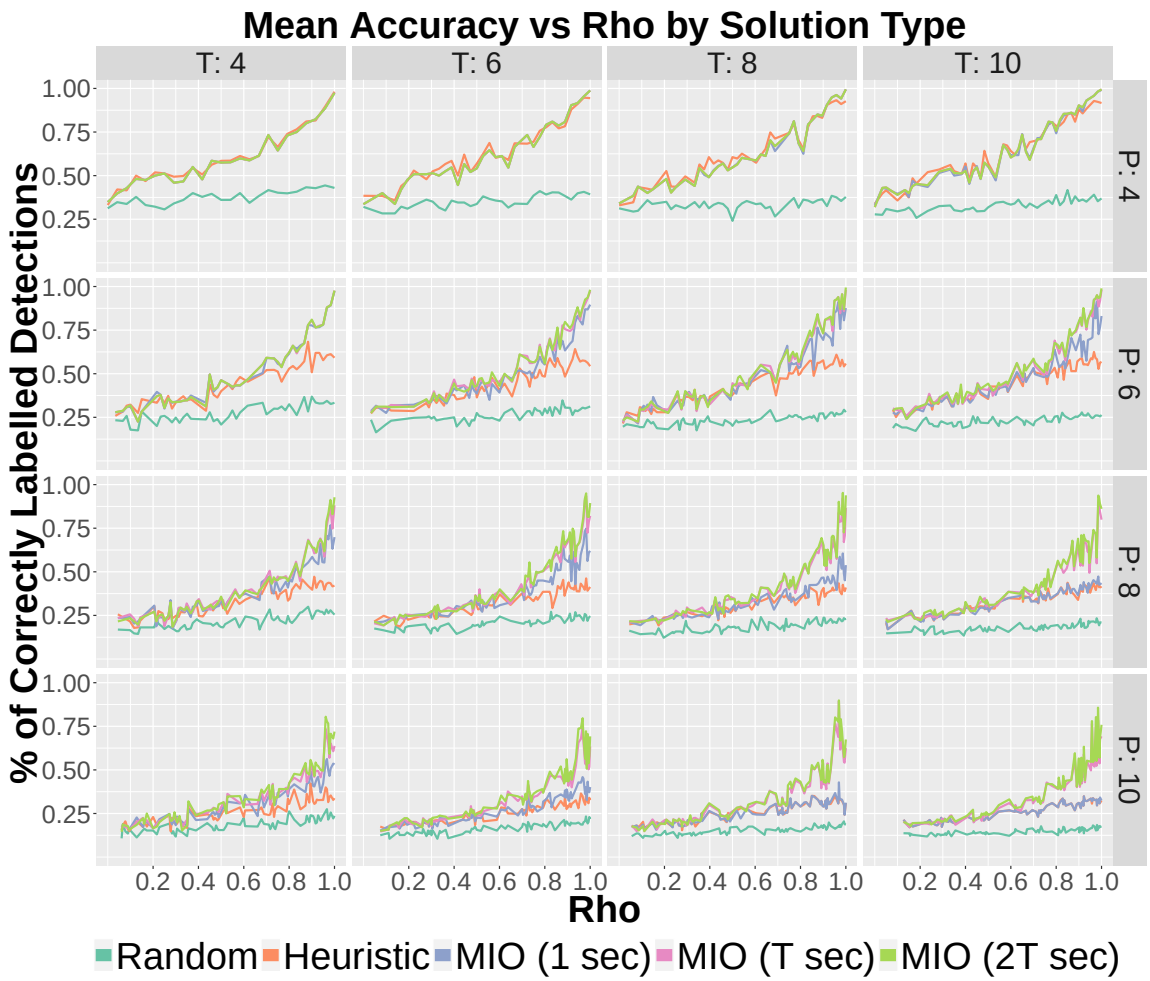

Figure 7-4: Accuracy of MIO compared against the heuristic and a randomized solution.

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.

As mentioned previously in regards to the heuristic, a sliding window algorithm would make decisions on a subset of scans, and these decisions will be fixed before

accepting a new set of scans. In regards to the MIO, this would be implemented by adding constraints to restrict the values of y_{itj} to match that of the subsetted solution. The fact that the MIO finds very good solutions in T or fewer seconds means that a sliding window algorithm would be able to solve each subset in real time before advancing to the next subset of scans. Furthermore, the MIO would likely benefit from the fixed decisions of the preceding windows, since this is added knowledge that has not utilized by our approaches.

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 7-5 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.

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

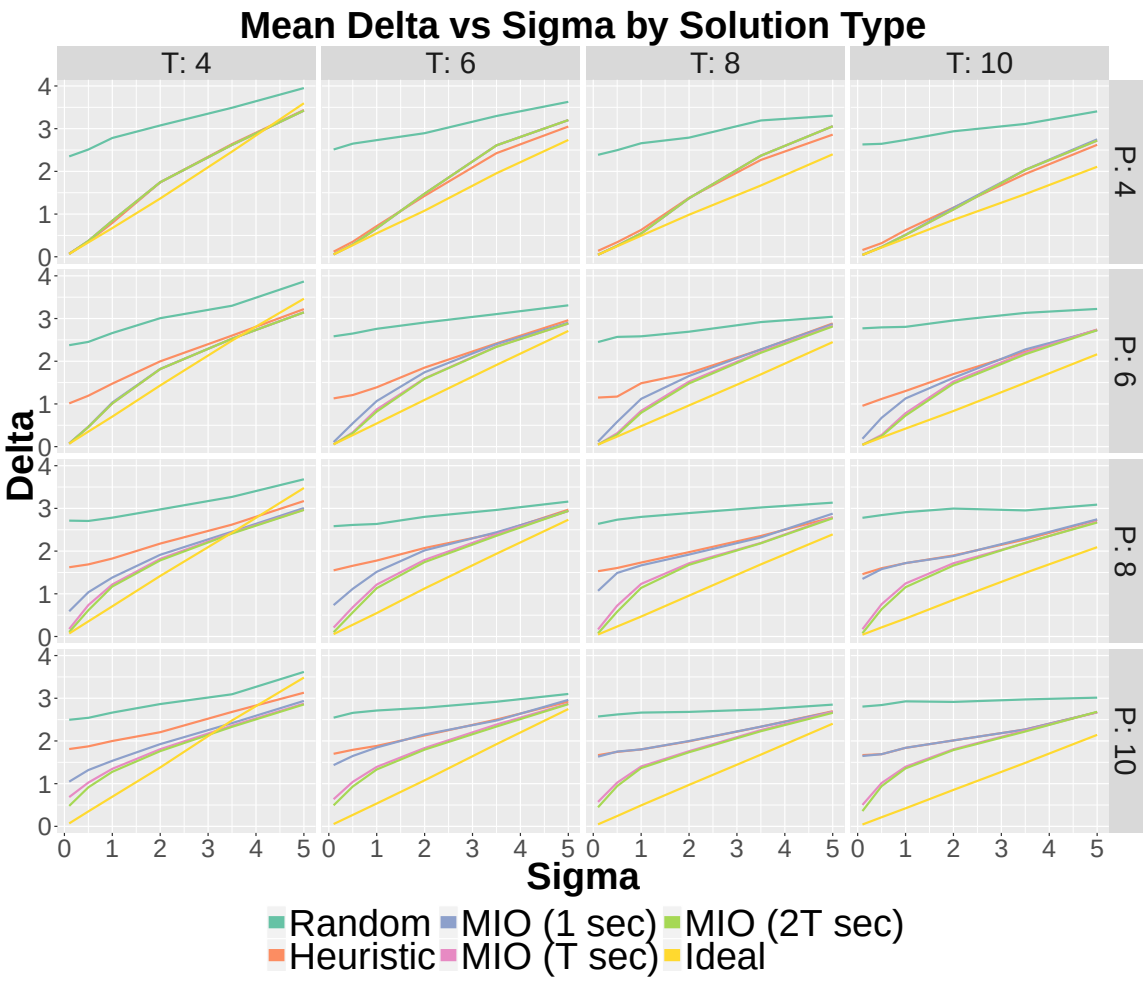

Figure 7-5: Trajectory estimation performance

from the ideal solution converges to the true ground track.

7.0.10 Robust Scenario

Here we extend the discussion from the previous section to analyze the performance of our methods on scenarios with detection ambiguity. We first summarize our experimental methods before discussing performance of both the robust heuristic and the robust MIO in the spheres of both the data association and trajectory estimation problems.

This experiment serves as an extension of the basic one, in order to test the performance of our algorithms under detection ambiguity. We use the same scenarios

generated from the basic experiment, but due to the additional difficulty inherent with detection ambiguity, we limit the range of signal noise to $\sigma \in \{0.1, 0.5, 1.0, 2.0\}$, excluding choosing to exclude the extreme cases of signal noise. In addition, we simulate both missed detections and false alarms. A detection is removed with probability, γ, and we consider $\gamma \in \{0.2, 0.15, 0.1, 0.05\}$. We do not allow empty scans. For each scan, we generate false alarms according to a poisson distribution with parameter, λ, and detection locations are then randomly selected uniformly within the state space. We consider $\lambda \in \{0.1, 0.5, 0.1, 2.0\}$. The false alarms are then added to $\mathcal{X}_t$ and the detection order of $\mathcal{X}_t$ is again randomly shuffled as before.

Once the data has been generated, we follow the same sequence as before in the basic experiment, running the heuristic first and then feeding the solution into the MIO as a warm start. Note that the heuristic is given 1,000 starting points only, as concluded from the results of the basic experiment. Once again, the optimization process was set to terminate after 3T seconds, with solutions collected at intervals of $\{1, T, 2T, 3T\}$ seconds. Prior to the running of this experiment, we performed a mini experiment and used the results to tune the penalties θ and ϕ. A summary of the exact penalties used along with an explanation of the insight behind them, can be found in Appendix B.

We will now evaluate the performance of the robust heuristic and MIO. We begin with a discussion on the run times of the robust heuristic, following the number of targets estimation, accuracy and trajectory estimation.

Robust Heuristic Run Times

Table 7.2 summarizes the minimum, mean, and maximum run times of the heuristic from Experiment 2 for a single starting point, arranged by the number of estimate targets ($P_{estimated}$) and number of scans (T). Times are shown in milliseconds.

Comparing Table 7.1 and Table 7.2 we see that the run times for an estimated number of targets in the robust heuristic range from roughly double to three and four times that of a comparable number of targets in the basic heuristic. Due to the increase in combinatorial solutions in the robust heuristic over the basic heuristic, this

$P_{\text{estimated}}$	T	Heuristic Run Times (in milliseconds) Min	Mean	Max
2	4	0.15	0.23	0.41
2	6	0.42	0.56	0.93
2	8	0.77	1.04	2.24
2	10	1.27	1.73	20.23
4	4	0.15	0.34	1.04
4	6	0.50	0.94	2.69
4	8	1.09	1.88	3.87
4	10	2.12	3.25	13.51
6	4	0.14	0.42	0.96
6	6	0.57	1.29	4.45
6	8	1.33	2.66	75.28
6	10	2.53	4.61	18.69
8	4	0.16	0.50	1.10
8	6	0.60	1.59	3.46
8	8	1.38	3.37	6.87
8	10	2.63	5.84	12.40
10	4	0.18	0.55	1.10
10	6	0.72	1.82	3.98
10	8	1.53	3.96	8.18
10	10	3.42	6.93	13.93
12	4	0.16	0.56	0.99
12	6	0.99	1.95	3.96
12	8	1.74	4.33	8.69
12	10	3.40	7.71	15.10

Table 7.2: Robust heuristic run times (in milliseconds) for a single starting point.

was an expected result. The robust heuristic, however, can be parallelized in the same many as the basic heuristic, meaning that the robust heuristic can actually recover the speed of the basic heuristic with the introduction of additional processors. More importantly, we see that the robust heuristic scales very efficiently with $P_{\text{estimated}}$, and this the scaling actually improves as the number of estimated targets and scans increases. Increasing from two to six estimated targets for four scans roughly triples the run time, while increasing from eight to twelve estimated targets for four scans increases the run time by only 12%. This is a great result because it means that a relatively wide range of estimated targets can be parallelized without fear of one subset requiring a substantially longer run time than another.

Although the robust heuristic scales well with the number of estimated targets, it is more sensitive to increases in the number of scans. However, this effect is no worse than what we saw for the basic heuristic. It appears that on average increasing the number of scans by 2 results in a doubling of the run time for a fixed number of estimated targets. We conclude that these results again support the use of the robust heuristic in an online algorithm with a sliding window, as discussed in the previous section.

Evaluating the Number of Targets

Next, we continue our analysis of the robust approaches by quantifying the algorithms' ability to estimate the correct number of targets. This is perhaps the most important goal of a MTT algorithm and so we begin our analysis here. To this end, we define

$$P_{\text{difference}} = P_{\text{true}} - P_{\text{estimated}} \tag{7.2}$$

where $P_{\text{estimated}}$ is the number of estimated targets and P_{true} is the number of true targets, and we plot the distribution of $P_{\text{estimated}}$. Figure 7-6 shows the distribution of $P_{\text{difference}}$ for scenarios with four targets and eight scans, and for comparison, Figure 7-7 plots the same result for scenarios of eight targets and eight time scans.

Note that the algorithms have correctly estimated the number of targets when $P_{\text{difference}} = 0$. When $P_{\text{difference}} < 0$, we have overestimated the number of targets, and when $P_{\text{difference}} > 0$, we have underestimated the number of targets. We see that both the robust heuristic and the robust MIO estimate the number of targets correctly a high proportion of the time in the scenario with four targets, particularly for smaller values of λ. As λ increases, though, both algorithms tend to underestimate. The same trend persists in the larger scenario. This suggests that either 1) the false alarm penalty needs further tuning and likely was not set high enough in the experiment or 2) the missed detection penalty set too high. In the case where θ is set too low, the algorithms would prefer to classify detections as false alarms rather than

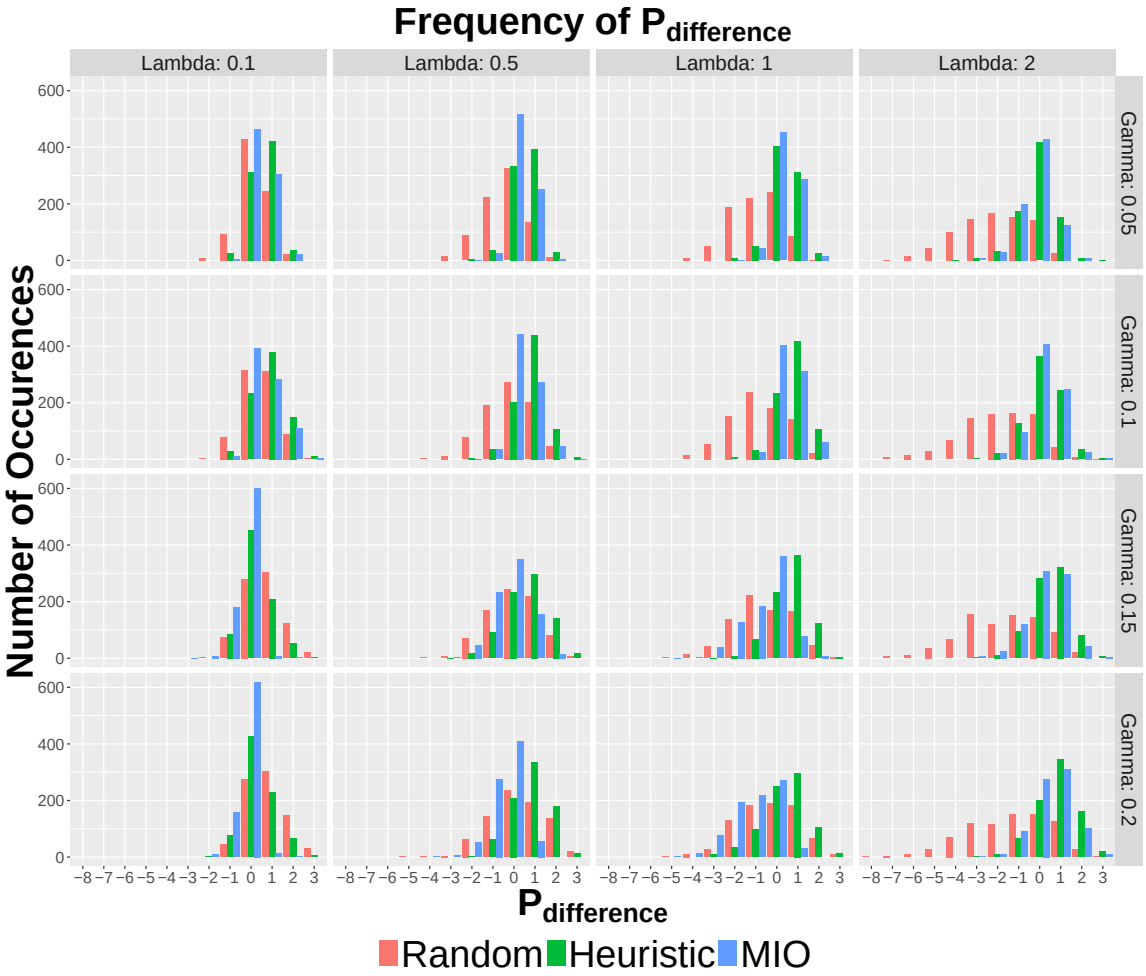

Figure 7-6: Distribution of the difference in true and estimated number of targets for scenarios with 4 targets and 8 scans, arranged by γ and λ.

create additional trajectories for the detections. In the case of ϕ set too high, the algorithms would opt out of creating additional trajectories in order to decrease the need to fill smaller scans with missed detections. Furthermore, because the effect of underestimation is more prominent in the scenario with more targets, we conclude that both penalties should probably take into account the number of targets that it is currently estimating.

Data Association

Knowing that we tend to underestimate the number of targets with the given penalties, we move on in our analysis to measuring the accuracy of our robust ap-

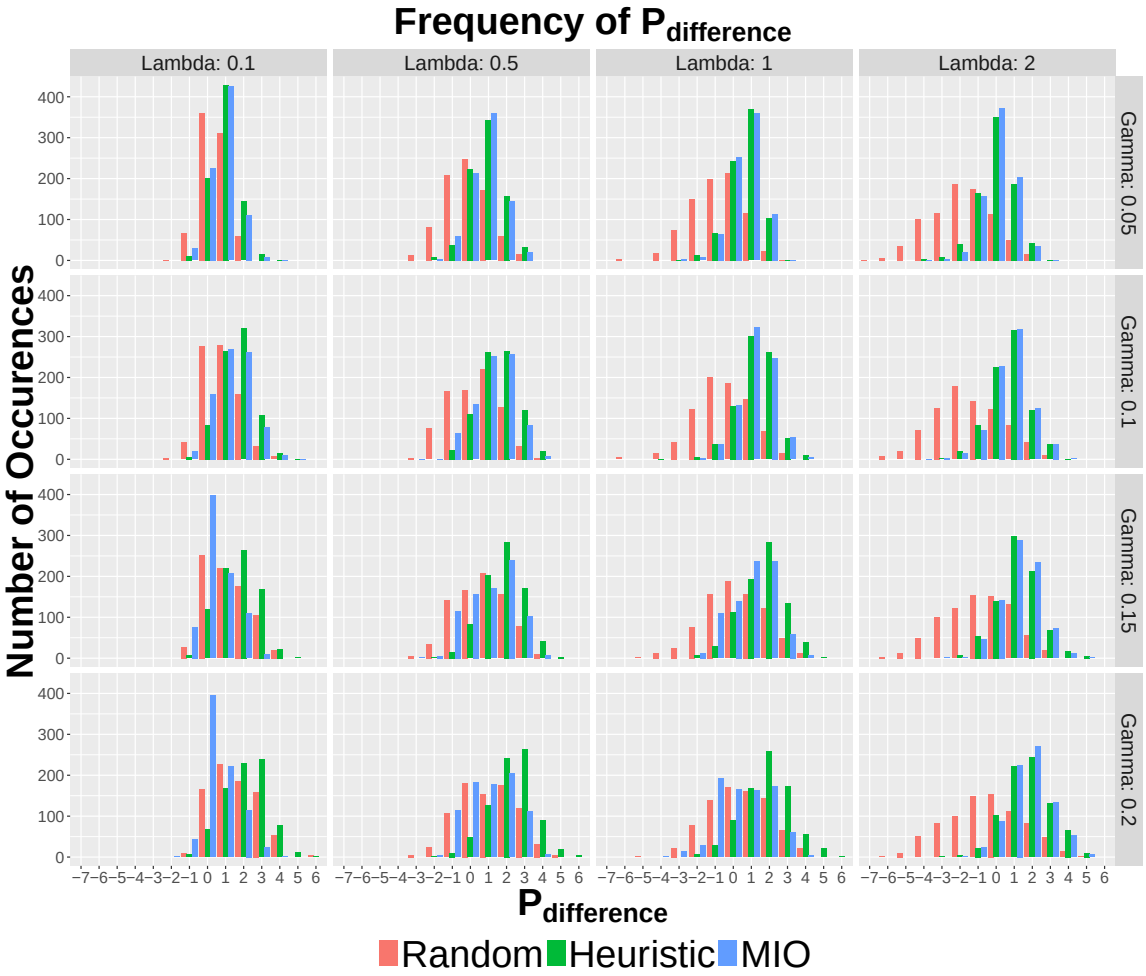

Figure 7-7: Distribution of the difference in true and estimated number of targets for scenarios with 8 targets and 8 scans, arranged by γ and λ.

proaches. Figures 7-8 and 7-9 plot the accuracy performance metric against the difficulty metric, ρ, for scenarios of four and eight targets, respectively. Both scenarios have eight scans and both Figures have been arranged by γ and λ.

Similar to the performance of the basic heuristic, we again see that the robust heuristic improves greatly over that of a random solution, and the MIO offers even further improvement. Again, we seeing that running the MIO for 1 second offers significant improvements over the heuristic, and running the MIO for T seconds offers further improvement. However, running the MIO for 2T seconds offers little to no further improvement. Again, these results support the use of the MIO in an online algorithm with a sliding window as mentioned previously.

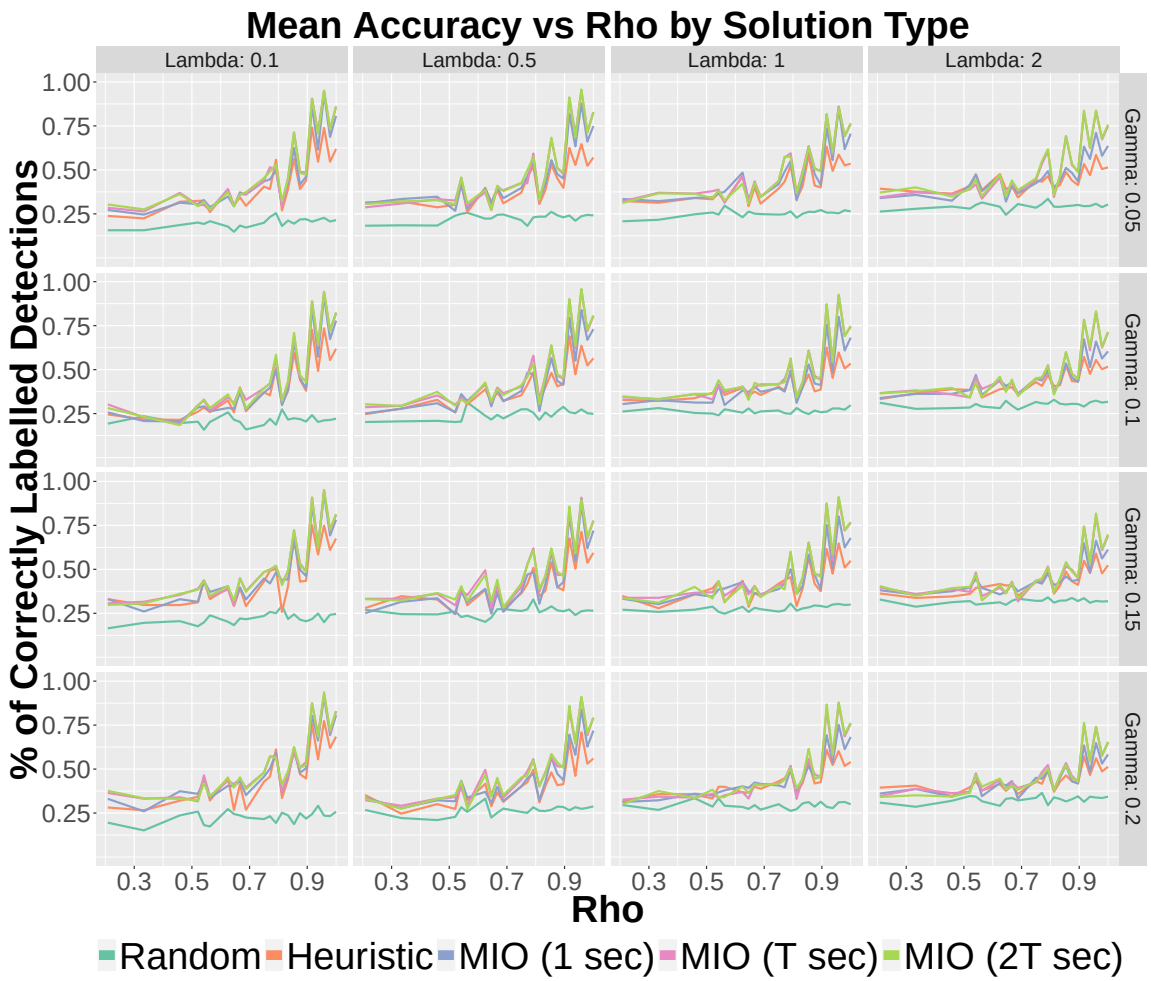

Figure 7-8: Accuracy of robust heuristic and MIO as compared to random solutions for scenarios of 4 targets and 8 scans, arranged by γ and λ.

Comparing Figure 7-8 with the 4 target and 8 scan element of Figure 7-4, we see only a slight decrease in performance when $\gamma = 0.05$ and $\lambda = 0.1$. This is an important result because we no longer know the number of targets in the robust case, yet we achieve almost the same levels of accuracy. Furthermore, both Figure 7-8 and Figure 7-9 show that the robust algorithms are more robust to decreases in the detection probability γ than to increases in the false alarm rate λ. We conclude that the robust approaches are more sensitive to changes in the false alarm rate, in particular when it comes to making data associations.

We have shown that both the heuristic and the MIO tend to underestimate the number of targets, due to the chosen penalties. We have also shown that accuracy de-

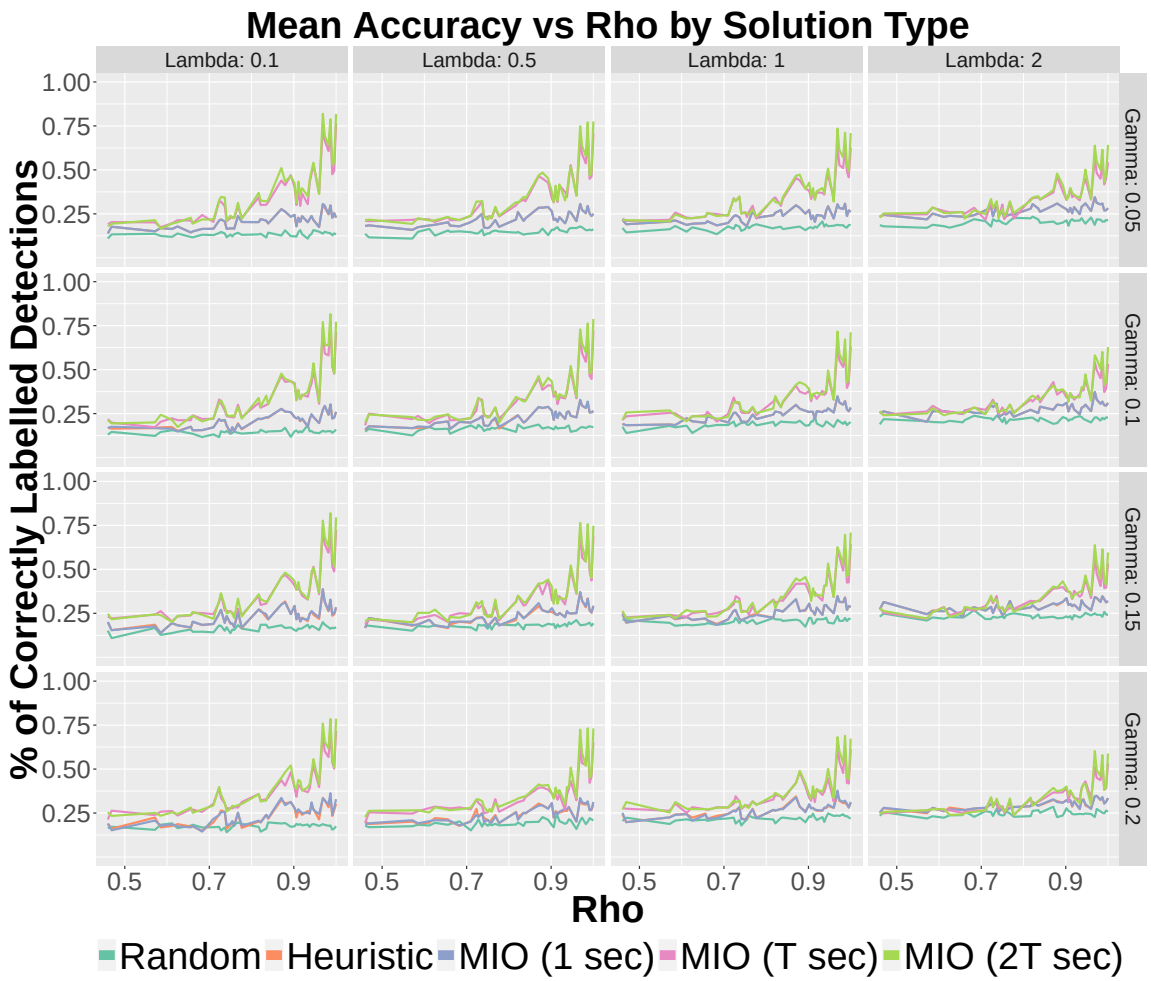

Figure 7-9: Accuracy of robust heuristic and MIO as compared to random solutions for scenarios of 8 targets and 8 scans, arranged by γ and λ.

grades as the false alarm rate increases. This is likely not a coincidence. It is probable that as a result of underestimation, in which fewer trajectories are generated, there is a higher rate of misclassification of detections as false alarms, which in turn directly leads to a reduced accuracy. Therefore, it is a promising result to see accuracies above 75% in Figure 7-9, even when Figure 7-7 suggests overestimation. It is entirely possible that further parameter tuning or introducing more complex penalties would lead to even great performance in the data association problem.

Trajectory Estimation

We conclude our analysis of the robust approaches with a discussion on their performance in the sphere of the trajectory estimation problem. Figures 7-10 and 7-11 plot the δ performance metric against the difficulty metric, σ, for scenarios of four and eight targets, respectively. Again, both scenarios have eight scans and both Figures have been arranged by γ and λ. Note that the range on σ has been reduced from $[0.1, 5.0]$ to $[0.1, 2.0]$.

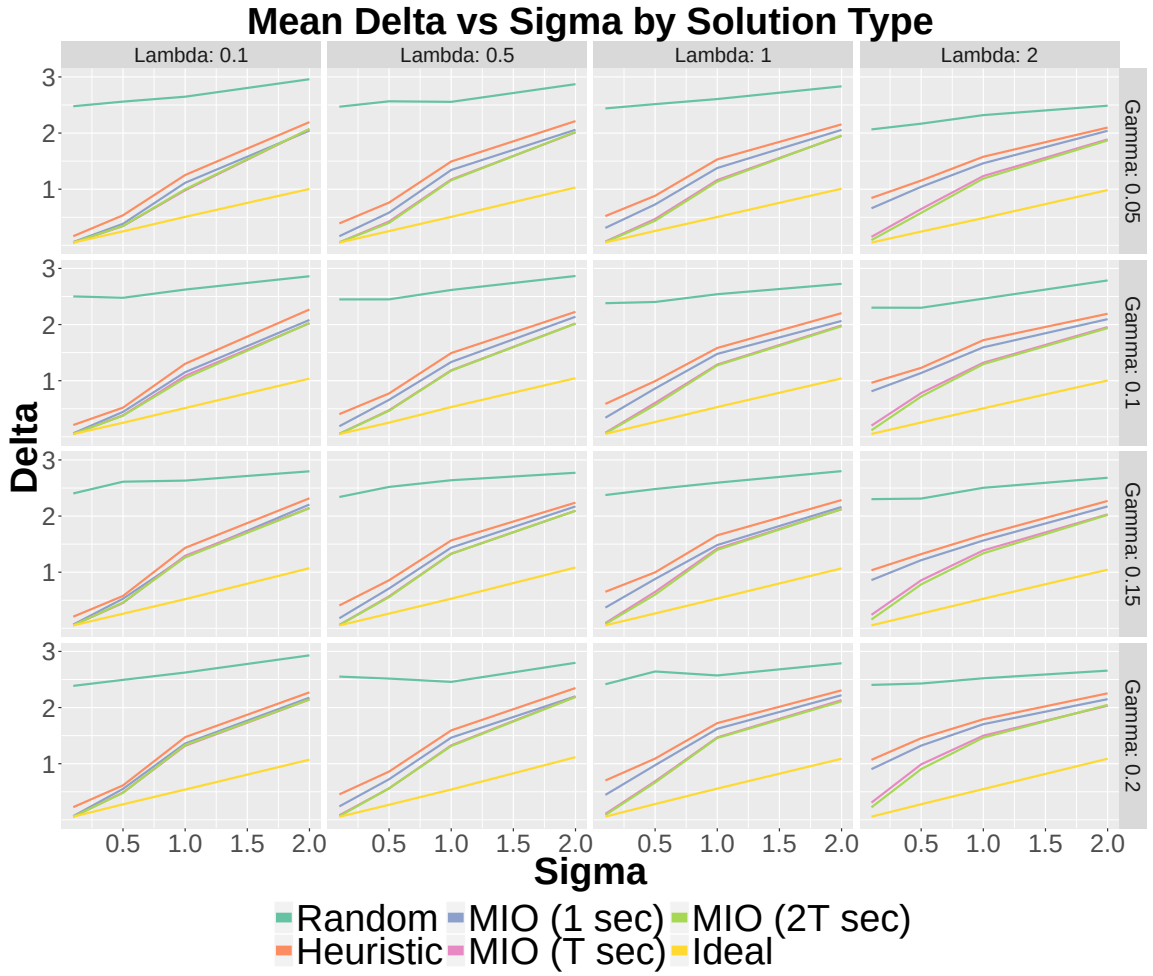

Figure 7-10: δ of robust heuristic and MIO as compared to random solutions for scenarios of 4 targets and 8 scans.

Again, we measure against the basic approaches by comparing Figure 7-10 with the 4 target and 8 scan element of Figure 7-5. We see that the robust approaches do not drastically reduce in performance for the easiest robust scenario of $\gamma = 0.05$

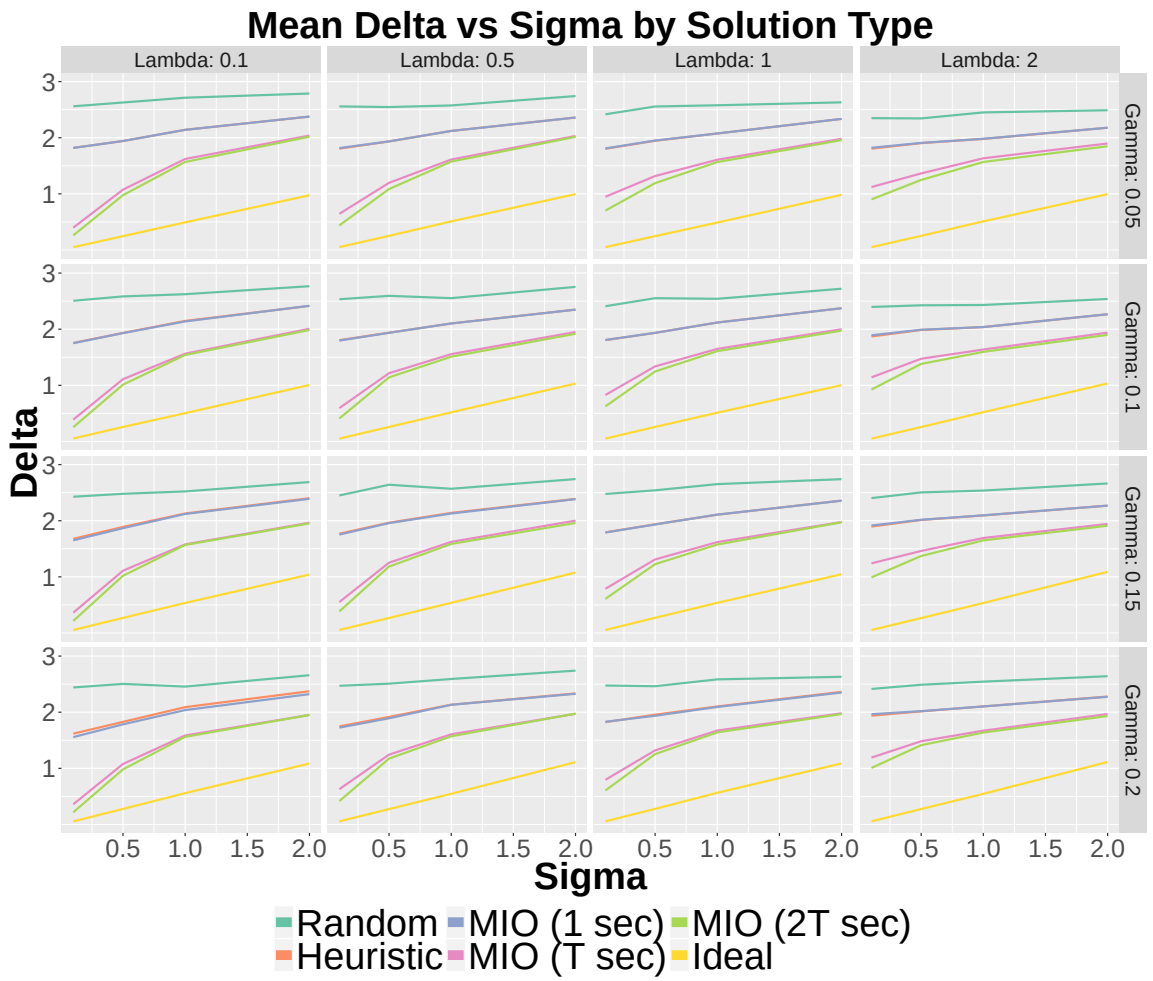

Figure 7-11: δ of robust heuristic and MIO as compared to random solutions for scenarios of 8 targets and 8 scans.

and $\lambda = 0.1$. However, the gap in performance between the ideal solution and the solutions of the robust algorithms grows wider with increases in σ, something that is expected but was not as sizable in the basic experiment. Therefore, the robust approaches may be less robust to increases in σ in scenarios with detection ambiguity. However, it also appears that these methods are more robust to increases in the false alarm rate λ when it comes to trajectory estimation than they were when it came to data association, especially in the larger scenario shown in Figure 7-11. We conclude that our robust methods are fairly robust to both increases in the false alarm rate and decreases in trajectory estimation under detection ambiguity, but increasing the signal noise degrades the performance of our methods more so in scenarios with detection

ambiguity than in scenarios without detection ambiguity.

7.0.11 Conclusion

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 8

Summary and Future Work

We presented multi-target tracking approaches which jointly solve the problems of data association and trajectory estimation via global optimization methods using a single objective function. To this end, we proposed the use of a randomized local search heuristic as a warmstart for a mixed integer optimization model, and we did so for scenarios with and without detection ambiguity. We accomplish this without the need of a trajectory bank nor the a prior computation of trajectory hypotheses. We showed that the heuristic finds good quality feasible solutions very quickly. We also show that the MIO offers improvement over the heuristic and this improvement is found in real time for applications considered by this paper. The proposed methods show potential for use in an online algorithm with a sliding window, suggesting one possible extension for future work. Other extensions for future work could seek to expand the formulations to account for the birth/death of targets. Additionally, future research could explore the use of more complex penalty functions. One possible idea would be the use of piecewise linear functions which would require further expansion of our formulations.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Estimated Trajectory Assignments

In order to analyze the performance of a multi-target tracking algorithm, we must find the best matching of the true trajectories of the scenario and the estimated trajectories of the algorithm solution. Put differently, we wish to find a set of assignment pairings which match true and estimated trajectories. Here we present a linear optimization model which solves for the globally optimal assignment pairings of true and estimated trajectories. In addition, we discuss the required

The goal of this assignment problem is to optimally assign pairs of true trajectories i to estimated trajectories j if there exists such a pairing to be made. Only a single set of decision variables are needed to determine if the true trajectory i should be assigned to the estimated trajectory j or not.

$$y_{ij} = \begin{cases} 1, & \text{if true trajectory } i \text{ is assigned} \\ & \text{to estimated trajectory } j, \\ 0, & \text{otherwise.} \end{cases}$$

Remember that we denote the true position of trajectory i at scan t with $\bar{x}_{it}$ and the estimated position of trajectory j at scan t with $\hat{x}_{jt}$. Then the cost c_{ij} of assigning true trajectory i to estimated trajectory j is the norm distance between these two

trajectories as measured at each scan.

$$c_{ij} = \sum_{t=1}^{T} \|\bar{x}_{it} - \hat{x}_{jt}\| \tag{A.1}$$

If we denote the true number of targets as P_{true} and the estimated number of targets as $P_{\text{estimated}}$ then the objective of the integer optimization model would be:

$$\underset{y_{ij}}{\text{minimize:}} \sum_{i=1}^{P_{\text{true}}} \sum_{j=1}^{P_{\text{estimated}}} c_{ij} y_{ij} \tag{A.2}$$

When the number of true targets is equal to the number of estimated targets ($P_{\text{true}} = P_{\text{estimated}} = P$), we simply require two equality constraints to ensure that each true trajectory i is assigned to exactly one estimated trajectory j and vice versa.

$$\sum_{i=1}^{P} y_{ij} = 1 \qquad \forall j = 1, ..., P \tag{A.3}$$

$$\sum_{j=1}^{P} y_{ij} = 1 \qquad \forall i = 1, ..., P \tag{A.4}$$

However, when the number of estimated trajectories differs from the number of true trajectories, these constraints must be modified slightly. In the case where the number of true targets exceeds the estimated number of targets ($P_{\text{true}} \geq P_{\text{estimated}}$), we restrict each true trajectory i to the assignment of *at most* one estimated trajectory j, and Equation A.3 is modified to

$$\sum_{i=1}^{P_{\text{true}}} y_{ij} \leq 1 \qquad \forall j = 1, ..., P_{\text{estimated}} \tag{A.5}$$

On the contrary, when the number of estimated targets exceeds the true number of targets ($P_{\text{true}} \leq P_{\text{estimated}}$), then we restrict each estimated trajectory j to the

assignment of *at most* one true trajectory i, and Equation A.4 is modified to

$$\sum_{j=1}^{P_{\text{estimated}}} y_{ij} \leq 1 \qquad \forall i = 1, ..., P_{\text{true}} \tag{A.6}$$

In summary, the generalized integer optimization assignment model is presented below.

$$\begin{aligned}
\underset{y_{ij}}{\text{minimize:}} \quad & \sum_{i=1}^{P_{\text{true}}} \sum_{j=1}^{P_{\text{estimated}}} c_{ij} y_{ij} \qquad (A.7) \\
\text{subject to:} \quad & \sum_{i=1}^{P_{\text{true}}} y_{ij} = 1 \qquad \forall j = 1, ..., P_{\text{estimated}} \\
& \sum_{j=1}^{P_{\text{estimated}}} y_{ij} = 1 \qquad \forall i = 1, ..., P_{\text{true}} \\
& y_{ij} \in \{0, 1\} \quad \forall i = 1, ..., P_{\text{true}}, j = 1, ..., P_{\text{estimated}}
\end{aligned}$$

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Experiment 2 Penalty Values

Here we provide recommendations for the tuning of penalty parameters θ and ϕ. We begin with an explanation of grounded in logic. It can be shown that as the false alarm rate λ increases, the number of expected false alarms also increases. Therefore, it stands to reason that as a general rule of thumb the false alarm penalty θ should decrease as λ increases. Similarly, the number of expected missed detections increases as the missed detection probability γ increases, and so too the missed detection penalty should decrease. Then it follows logically that the missed detection penalty ϕ should increase as γ decreases. Furthermore, it is convenient to reason that the value of both of these penalties should somehow be tied to the value of σ, though this is a more difficult sequence of logic to justify. Through examination we found these logical concepts to hold true across a variety of scenario sizes and difficulties. Using this insight as well as the results of a mini experiment, we arrived at the penalty values summarized in Table B.1.

σ	γ	λ	θ	ϕ
0.1	0.2	0.1	0.4	0.1
0.1	0.2	0.5	0.2	0.1
0.1	0.2	1.0	0.3	0.2
0.1	0.2	2.0	0.1	0.2
0.1	0.15	0.1	0.4	0.1
0.1	0.15	0.5	0.4	0.1
0.1	0.15	1.0	0.4	0.3
0.1	0.15	2.0	0.1	0.3
0.1	0.1	0.1	0.3	0.1
0.1	0.1	0.5	0.3	0.2
0.1	0.1	1.0	0.2	0.2
0.1	0.1	2.0	0.1	0.3
0.1	0.05	0.1	0.3	0.2
0.1	0.05	0.5	0.2	0.2
0.1	0.05	1.0	0.1	0.4
0.1	0.05	2.0	0.1	0.4
0.5	0.2	0.1	0.5	0.1
0.5	0.2	0.5	0.4	0.2
0.5	0.2	1.0	0.4	0.2
0.5	0.2	2.0	0.3	0.3
0.5	0.15	0.1	0.5	0.1
0.5	0.15	0.5	0.4	0.2
0.5	0.15	1.0	0.4	0.3
0.5	0.15	2.0	0.3	0.4
0.5	0.1	0.1	0.4	0.3
0.5	0.1	0.5	0.4	0.3
0.5	0.1	1.0	0.3	0.3
0.5	0.1	2.0	0.2	0.4
0.5	0.05	0.1	0.4	0.4
0.5	0.05	0.5	0.4	0.4
0.5	0.05	1.0	0.3	0.5
0.5	0.05	2.0	0.3	0.5
1.0	0.2	0.1	0.5	0.1
1.0	0.2	0.5	0.5	0.2
1.0	0.2	1.0	0.5	0.1
1.0	0.2	2.0	0.5	0.4
1.0	0.15	0.1	0.5	0.1
1.0	0.15	0.5	0.5	0.3
1.0	0.15	1.0	0.5	0.2
1.0	0.15	2.0	0.5	0.4
1.0	0.1	0.1	0.5	0.5
1.0	0.1	0.5	0.5	0.4
1.0	0.1	1.0	0.4	0.4
1.0	0.1	2.0	0.4	0.5
1.0	0.05	0.1	0.5	0.5
1.0	0.05	0.5	0.5	0.5
1.0	0.05	1.0	0.5	0.5
1.0	0.05	2.0	0.5	0.5
2.0	0.2	0.1	0.5	0.1

Bibliography

[1] 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*, pages 1926–1933, June 2012.

[2] Anton Andriyenko and Konrad Schindler. Multi-target tracking by continuous energy minimization. In *CVPR*, 2011.

[3] Y. Bar-Shalom and X.R. Li. *Multitarget-multisensor Tracking: Principles and Techniques*. Yaakov Bar-Shalom, 1995.

[4] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[5] Robert 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.

[6] S.S. Blackman. *Multiple-target Tracking with Radar Applications*. Radar Library. Artech House, 1986.

[7] S.S. Blackman. Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1):5–18, Jan 2004.

[8] Nadya Bliss, Robert Bond, Jeremy Kepner, Hahn Kim, and Albert Reuther. Interactive grid computing at lincoln laboratory. *MIT Lincoln Laboratory Journal*, 16(1), 2006.

[9] Craig Carthel and Stefano Coraluppi. Multi-hypothesis sonar tracking. In *Fusion 2004: Seventh International Conference on Information Fusion*, 2004.

[10] D. Castanon. Efficient algorithms for finding the k best paths through a trellis. *IEEE Transactions on Aerospace and Electronic Systems*, 26(2):405–410, Mar 1990.

[11] Inc. Gurobi Optimization. Gurobi 6.5 performance benchmarks. `http://www.gurobi.com/pdfs/benchmarks.pdf`, 2015. Accessed 4 April 2016s.

[12] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015.

[13] C. Heij, P. de Boer, P.H. Franses, T. Kloek, H.K. van Dijk, and A.E.U. Rotterdam. *Econometric Methods with Applications in Business and Economics*. OUP Oxford, 2004.

[14] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[15] Andrea Lodi. *50 Years of Integer Programming 1958-2008.From the Early Years to the State-of-the-Art*. Springer Berlin Heidelberg, 2010.

[16] Miles Lubin and Iain Dunning. Computing in operations research using julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.

[17] C.L. Morefield. Application of 0-1 integer programming to multitarget tracking problems. *Automatic Control, IEEE Transactions on*, 22(3):302–312, Jun 1977.

[18] George Nemhauser. Integer programming: Global impact. `https://smartech.gatech.edu/bitstream/handle/1853/49829/presentation.pdf`, 2013. Accessed 4 April 2016s.

[19] 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*, volume 1, pages 436–440 vol.1, Nov 1998.

[20] Aubrey P. Poore and Nenad Rijavec. A lagrangian relaxation algorithm for multidimensional assignment problems arising from multitarget tracking. *SIAM Journal on Optimization*, 3(3):544–563, 1993.

[21] G.W. Pulford. Taxonomy of multiple target tracking methods. *Radar, Sonar and Navigation, IEE Proceedings* -, 152(5):291–304, October 2005.

[22] D.B. Reid. An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on*, 24(6):843–854, Dec 1979.

[23] 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*, 59(7):3452–3457, July 2011.

[24] N. Jakovcevic Stor, I. Slapnicar, and J. L. Barlow. Accurate eigenvalue decomposition of real symmetric arrowhead matrices and applications. *Linear Algebra and Its Applications*, 464:62–89, 2015.

[25] Top500 Supercomputer Sites. Performance development. `http://www.top500.org/statistics/perfdevel/`, 2015. Accessed 4 April 2016s.

[26] 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*, 25(2):287–296, Mar 1989.