Track and Truth Correlation in Military Simulations: Solution Methods within an Assignment Problem Framework

Siem van Benthem

Master thesis project in collaboration with TNO
Submitted by Siem van Benthem (4965124) on June 14th, 2024
in fulfillment of the degree of Master Applied Mathematics

Supervised by David de Laat (TU Delft), Paul Eigeman (TNO),
Nicole van Elst (TNO) and Kjeld Broekema (TNO)

Defense on the 26th of June 2024

Applied Mathematics - Discrete Mathematics and Optimization
Delft Institute of Applied Mathematics (DIAM)
Delft University of Technology

TNO department Modelling of Tactical Operations (MTO)
Den Haag Oude Waalsdorperweg

Abstract

Military simulation is essential for modern warfare, providing a virtual environment for training, analysis, and rehearsal of procedures. Accurate correlation between simulated entities, called tracks, and their radar detections, called tracks, is crucial for generating reliable data, vital for evaluating military operations and improving training exercises. However, factors like radar noise, communication errors, and simulation inaccuracies complicate this correlation. This thesis aims to develop a robust method for correlating simulated truth entities with corresponding radar tracks in military simulations. The proposed method tackles challenges such as radar noise and communication errors to improve the reliability and validity of simulation statistics. The research encompasses the theoretical development of three correlation algorithms, one of which serves as a benchmark for verification. The methods were evaluated across various simulated scenarios, in which the two correlation methods consistently outperformed the benchmark, particularly in scenarios with fewer data points.

Contents

1 Introduction

Military simulation is an essential tool in modern warfare, enabling the training, analysis, and rehearsal of military procedures in a virtual environment. These simulations rely heavily on the accurate modeling and representation of realistic systems and scenarios, among which are the detection and tracking of entities by radar systems. The correct correlation of simulated entities, referred to as *truths*, with their detections by radar systems, called *tracks* or *perceived truths*, is essential for generating reliable data and statistics for simulation exercises. This data is fundamental for assessing the performance of military operations and providing insights during and after training exercises.

In a typical military simulation, truth entities represent units' actual positions and identities within the simulated environment. Simulated radar systems detect these entities and generate tracks based on the received signals. However, various factors complicate the correlation between truth entities and radar tracks. As proclaimed by experts, radar noise, communication and network errors, datalink protocol faults, and inherent simulation inaccuracies can lead to discrepancies between the actual positions of entities and the positions indicated by radar tracks.

The ability to correctly correlate truth entities with radar tracks has significant implications for military training and operations. Reliable correlation ensures that simulation statistics, such as detection rates, detection times, and tracking accuracy, reflect true performance. These statistics are vital for evaluating the effectiveness of strategies and tactics and identifying areas that require improvement.

The primary objective of this thesis is to develop a robust method for correlating simulated truth entities with the corresponding radar tracks in a military simulation environment. This method will address the various challenges posed by radar noise, communication errors, and simulation inaccuracies. By achieving accurate correlation, the proposed method aims to enhance the reliability and validity of the statistics generated during training exercises.

The scope of this thesis encompasses the theoretical development of the correlation algorithm, its practical implementation, and the evaluation of its effectiveness in various simulated scenarios. While the focus is on simulated radar systems, the principles and techniques developed may also apply to real-world radar systems with appropriate modifications.

At first, Section 2 familiarizes the reader with the required background on military simulation, as well as the important simulation exercise Joint Project Optic Windmill (JPOW) and simulation environment Joint Research on Air Defense Systems (JROADS). Subsequently, the problem of identifying a precise track and truth correlation is modeled using its practice-inspired properties in Section 3. In Section 4, the reader is introduced to several variations of the assignment problem, which will serve as the mathematical basis of

the novel solution method presented in Section 5. Verification of the solution method will be treated in Section 6, along with the corresponding results. In conclusion, the results are analyzed and reflected upon in Section 8.

2 Background on military simulation

In the modern military, computer-based simulation is an essential methodology aiming to replicate the complexities of warfare in virtual environments. It harnesses computational algorithms and advanced modeling techniques to provide soldiers, commanders, and strategic planners with realistic and interactive training experiences. From tactical engagements to strategic decision-making, computer-based simulations offer a dynamic platform for training skills, testing strategies and capabilities, and enhancing operational procedures in a risk-free environment.

The evolution of computer-based military simulation traces its roots to the early days of computing, where basic war games and simple simulations laid the groundwork for more sophisticated applications. With the improving quality of hardware and software, military simulations have evolved into highly immersive and realistic training environments. These simulations can range from individual training modules focusing on specific skills to large-scale operations involving multiple units.

Computer-based military simulation may be categorized into two types. At first, there is a hardware-in-the-loop simulation (HIL), which incorporates existing hardware in the simulation. The advantage of HIL-simulation is that real-life operating systems are utilized, therefore creating a more faithful representation of live events. Alternatively, all systems and equipment can be modeled digitally, yielding a simulation environment that does not use any existing systems. Such simulations are not named specifically, the context should clarify the type of simulation.

At the heart of military simulation lies a diverse array of software platforms and modeling tools. These technologies enable military organizations to recreate a wide range of battlefield conditions, such as urban warfare and asymmetric conflicts. By incorporating as many real-life aspects as possible, computer-based simulations strive to provide trainees with a comprehensive understanding of the complexities and uncertainties inherent in modern warfare.

One of the primary advantages of computer-based military simulation is its scalability and flexibility. Simulations can be designed to meet the specific training objectives and operational requirements of different military units, from individual soldiers to the entirety of forces in a conflict. Moreover, simulation-based experiments and exercises and can be easily modified and updated to reflect changes in tactics and technologies, ensuring that training remains relevant and up-to-date in an ever-changing environment.

Furthermore, computer-based military simulation offers an attractive alternative to traditional live exercises. By reducing the need for expensive equipment and logistical support, simulations enable military organizations to conduct training exercises more frequently and efficiently. Additionally, scenarios that are impossible to exercise live may be trained through military simulation. Moreover, simulations can be

conducted in controlled environments, mitigating risks to personnel and equipment while still providing realistic training scenarios that challenge participants.

However, the effectiveness of military simulation depends upon the fidelity and accuracy of the underlying models and algorithms. Ensuring that simulations accurately replicate the complexities of real-world conflicts, remains a key challenge for developers.

2.1 Overview of the datalink Link-16

In military applications, tactical data links (TDL) exist to ensure efficient and secure communication between allied forces during operations. The most widely used tactical data link by NATO (North Atlantic Treaty Organization) is Link-16, which started development in the 1970s by the United States Military [7]. This Section aims to explore the properties of Link-16 that are fundamental to the problem presented in Section 3.

The fundamental of Link-16 is its architecture i.e. the protocols governing data transmission. The architecture of Link-16 supports data transmission in a fixed format. Given that a unit will transmit information at a designated time, it can choose from a fixed set of messages. These messages are called J-series messages [7], each of which has several fields to be filled out by the sender. Each of these messages is labeled as Jx.y, where J dictates the TDL used (Link-16) and x.y the category and sub-category of the message respectively. For example, the message J3.2 is chosen for the transmission of positional data of airborne objects, such as enemy fighter jets and helicopters. There are many J-series messages, though the important ones for this thesis are those containing positional information and track management. These are summarized in Table 1. In practice, a unit can also send messages that are different from the J-series. However, for this thesis, these J-series messages are sufficient.

Due to limited capacity on the network, Link-16 enforces regulations on when a unit can transmit or receive data via the data link. The architecture of Link-16 adopts the principle of Time Division Multiple Access (TDMA) [7], a simple method allowing a single frequency channel to be used by multiple participants. In TDMA, each participant of the network is assigned a set of time slots, during which the unit can either transmit or receive data. Therefore, all participants consecutively send messages on the same frequency channel, allowing multiple participants to operate it. Often the time slot allocation is cyclic, meaning that the sequence of participants will repeat itself.

Label	Message Title	Purpose
J3.2	Air Track	Used to exchange information on air tracks.
J3.3	Surface Track	Used to exchange information on surface tracks.
J3.4	Subsurface Track	Used to exchange information on subsurface tracks.
J3.5	Land Point / Track	Used to exchange tactical information on land points and tracks.
J3.6	Space Track	Exchanges information on space and ballistic missile tracks.
J7.0	Track Management	Transmits information on track management (dropping tracks, identity conflicts, etc.)
J7.2	Correlation	Used to resolve dual designation by retaining one track and dropping the other.

Table 1: *J-series messages containing positional or management information [7].*

2.2 Distributed Interactive Simulation

Distributed Interactive Simulation (DIS) is a simulation (for any type of simulation) infrastructure developed by IEEE (Institute of Electrical and Electronics Engineers) to simulate realistic and complex virtual worlds [10]. This infrastructure allows different systems and technologies to be linked together in a simulation environment. In military simulations, DIS is the architecture of the simulation. Comparable with Link-16, DIS iteratively sends updates regarding the states of objects in the simulation through Protocol Data Units (PDUs). These PDUs, similar to J-series messages, have specific structures of fields filled out based on the entity type. The most crucial type of PDU is the entity state PDU (ePDU), which contains geographical data such as position, velocity, acceleration, and orientation. An important difference between Link-16 and DIS is the update frequency, DIS PDUs are generally sent more frequently.

2.2.1 Dead Reckoning

A fundamental aspect of DIS is dead reckoning, which refers to methods and protocols used to accurately predict the future positions of simulated entities. Each simulated entity has a dead reckoning model (DRM) that dictates how to estimate its future state. There are a total of eight DRMs, each of which is described by a three-letter abbreviation. The first letter, either F (Fixed) or R (Rotating), declares whether the orientation and rotation of the entity should be taken into account in the prediction. The second letter, one of P (Position) and V (Velocity), discloses if a first or second-order approximation for the position is used. The last letter, W (World) or B (Body), dictates in which coordinate system the dead reckoning is performed. The world coordinate system (W) is the general coordinate system used in the simulation and the body axis is centered at the entity [10]. The two most commonly used DRM models are the Fixed Position World (FPW) and Fixed Velocity World (FVW) models. The FPW model predicts the future position using the last known position and velocity. More precisely, the predicted position of an entity with last known position P_0 and velocity V_0 at time t_0, at time $t \geq t_0$, is

$$P = P_0 + V_0\,\Delta t, \qquad \Delta t = t - t_0$$

The FVW also takes into account acceleration A_0 at t_0, computing the future position via

$$P = P_0 + V_0\,\Delta t + \frac{1}{2} A_0\,\Delta t^2.$$

These two models are summarized in Table 2.

It is crucial to comprehend the circumstances under which entity state Protocol Data Units (PDUs) are transmitted. PDUs are transmitted for one of two primary reasons. Firstly, PDUs are sent periodically based on a predefined period. For instance, if the period is set to exactly one second, a new PDU is dispatched every second. Secondly, PDUs are transmitted when a precision threshold for dead reckoning is violated. If the difference between the predicted position of the entity according to the dead reckoning model and its actual position exceeds the set threshold, a new PDU is transmitted to update the position. In summary, whenever a PDU is sent, it is either because the predetermined period has elapsed or because the threshold for dead reckoning has been surpassed. These mechanisms ensure timely updates of entity positions and maintain accuracy.

DRM	Formula	Description
FPW	$P = P_0 + V_0\,\Delta t$	Constant velocity (or low acceleration) linear motion.
FVW	$P = P_0 + V_0\,\Delta t + \frac{1}{2} A_0\,\Delta t^2$	High speed (e.g., missile) or maneuvering at any speed.

Table 2: *Two dead reckoning models used in Distributed Interactive Simulation (DIS) [10].*

2.3 Joint Project Optic Windmill

Military simulation serves an especially crucial role in the development, preservation, continuation and practice of Integrated Air and Missile Defense (IAMD). The NATO Integrated Air and Missile Defence (NATO IAMD) is the the name given to the collection of assets and systems involved in the defense of NATO-occupied territory and population from air threats [15]. Naturally, protecting against such threats can be extremely complex due to the need to coordinate a large number of systems, resulting in numerous challenges. Military simulation is a powerful tool in exercising these challenges.

Out of all the challenges associated with NATO IAMD, the most significant is the harmonization and synchronization of all available NATO air-defense units [15]. Fundamental to this challenge is the issue of interoperability, which is defined to be the capacity of NATO nations to collaborate seamlessly, efficiently, and effectively in pursuit of tactical, operational, and strategic goals [14]. To maintain, review, and improve the protocols and aspects associated with interoperability, training is required.

The biggest European training event that supports NATO IAMD employing military simulation is the

Joint Project Optic Windmill (JPOW). During this HIL simulation exercise, the focus is on interoperability, and in particular the communication and collaboration of various weapon systems [5]. It aims to practice IAMD protocols on systems in a HIL simulation environment [16]. During JPOW, participants respond to predetermined IAMD scenarios, that are simulated on a vast computer network connecting all the participants [5]. Throughout the exercise, the learning objectives of all participants are continuously monitored and evaluated using the data gathered. The scenarios are designed to exercise specific aspects of IAMD, which are inspired by potential future real-world threats.

2.4 Joint Research on Air Defense Systems

Joint Research on Air Defense Systems (JROADS) is a simulation environment developed by TNO, designed to support the development of solutions to Integrated Air and Missile Defense (IAMD) challenges. The JROADS library consists of numerous features, among which are faithful representations of air defense and radar systems, accurate modeling of aircraft, user-defined simulation scenarios, and comprehensive data link connectivity, to name a few out of many. Due to its diversity of features, JROADS is a common choice for experimentation, development, and analysis purposes.

3 Problem description

This section presents the problem to be addressed in this thesis. First, the problem is introduced within the context of military simulation, by means of an example. Subsequently, the mathematical modeling of the structure and entities in the simulation is outlined, leading to the mathematical formulation of the problem.

Let's illustrate the problem with an example. Imagine a simulation scenario featuring several entities: a frigate (blue) equipped with radar and an anti-air defense system, and two fighters engaged in a dogfight (one blue and one red). The colors indicate the faction of the entities, with blue representing friendly forces and red representing foes. Throughout the scenario, the frigate detects both fighters using its radar system and engages the enemy fighter with its anti-air defense, intending to destroy it. However, instead of eliminating the enemy fighter, the frigate's missiles wrongly destroy the friendly fighter.

Undoubtedly, an error has occurred, though its exact nature remains uncertain. What's evident is that there are only two plausible explanations for this outcome. Either the frigate mistakenly identified the friendly fighter as a foe (and vice versa), or the anti-air system wrongly targeted the friendly fighter due to a system malfunction. To ascertain which event occurred, we seek to determine whether the first explanation holds true – whether the frigate misidentified the friendly fighter as an enemy.

To answer this question, every reported track in the scenario needs to be assigned to a truth object. We refer to the collection of all such assignments as a *track-truth link*. The issue of finding a track-truth link for a scenario arises frequently in military simulation, predominantly during the analysis of scenarios containing an error for which faulty detection can be an explanation. An effective method for the determination of such a matching in these scenarios greatly benefits accurate and thorough analyses by reducing uncertainty.

3.1 Properties of a track-truth link

To establish an effective mathematical framework for determining a track-truth link, it is crucial to identify key properties that can serve as foundational assumptions. This section explores several properties of a track-truth link.

3.1.1 Uniqueness of track detection

The first property concerns the uniqueness of track detection, asserting that each track represents the detection of at most one truth object. This principle originates from the custom in military simulation, where tracks typically describe a single entity. While it is feasible for a track to correspond to multiple objects, it is not preferred. It is crucial, in motivating this assumption, to differentiate between two cases: tracks deliberately reporting multiple truths as one, and those that do not.

If a track intentionally represents multiple truths, it is associated with a parameter known as *strength*, determining the number of entities it represents. Such tracks are employed only when the precise locations of the entities are irrelevant, providing only a general indication of the cluster's position. It is reasonable to presume that no other tracks or truths are in proximity to this cluster; otherwise, they would be included in the same track with a higher strength. Consequently, associating the track with corresponding truths becomes straightforward, and it is unnecessary to consider this in the track-truth link determination process. Thus, it is safe to assume that each track corresponds to at most one truth in the subsequent methodology.

Alternatively, multiple truths may be incorrectly reported as a strength-one track (or at least a smaller strength). In such instances, the track-truth link method ideally associates it with all the corresponding truths. However, this would pose several mathematical challenges, which will be explained at a later stage. Therefore, in the methodology developed it is assumed that such a track can only correspond to one truth, ideally being one of the corresponding truths. Regardless of which corresponding truth (assuming it is one) the track is mapped to, the outcome is neither entirely correct nor false.

Lastly, it is important to acknowledge that not every track necessarily corresponds to a truth object, as errors in detection can lead to tracks not aligning with an entity. Such tracks are categorized as *ghost tracks*.

3.1.2 Guaranteed truth detection

Second is the property of guaranteed truth detection, which claims that every truth entity taken into account is detected at least once. This assumption is not straightforward as it occurs quite often that truths are not detected. For example, when the truth's position is outside of detection range or terrain blocks the electromagnetic radiation, both resulting in non-detection. However, in the correlation method presented in Section 5, this property will be beneficial. If a truth is not detected, the non-existence of a track corresponding to the truth implies that there is no need for this truth to be associated with a track. Hence it is redundant in the track-truth link computation method and we can assume that all truths correspond to at least one track.

3.1.3 Uniqueness of truth per observer

The third property concerns the uniqueness of truth per observer. In military simulations, it is not uncommon for a single truth to be reported multiple times under different track numbers, despite existing protocols in tactical data links to prevent this. However, such occurrences are exceptionally rare for a single observer. Therefore, it is reasonable to assume that no two tracks of the same observer correspond to the same truth. In the unlikely event that this does occur, the superfluous tracks may be categorized as ghost tracks.

3.1.4 Time dependency

The fourth property is that of time dependency. Essentially, time dependency states that the track-truth link may change in time. The points in time where the track-truth link changes are called *alteration times*. There are two principal reasons why the track-truth link can change, which we list here. Firstly, there is the occurrence of *identity swapping*, which is illustrated through an example. Consider two fighters (red), labeled D_1 and D_2, flying in formation which are observed by a frigate's radar system (blue) and subsequently reported under track numbers T_1 and T_2, respectively. Initially, the correct track-truth link is (T_1, D_1) and (T_2, D_2). However, if the fighters alter their formation and fly in proximity to one another, it may confuse the radar system, and it may mistakenly report truth D_1 as T_2 and D_2 as T_1. Hence, if such confusion takes place, the track-truth link is no longer (T_1, D_1) and (T_2, D_2), but (T_1, D_2) and (T_2, D_1).

Second, there are *track correlation* and *track decorrelation*. In the datalink Link-16, correlation is the general name given to assigning received information to information already existing in the database [7]. Correlation is especially important in processing new track information, for which there exist correlation protocols preventing (to a certain degree) that no redundant information exists within the data link. Two such protocols are important for the time dependency of the track-truth link, those of track correlation and - decorrelation. The track correlation protocol removes duplicate track numbers from the data link if they represent the same object, which is decided if they are sufficiently similar in terms of position, velocity, direction etcetera. Conversely, if a track number is believed to no longer accurately represent an entity, the track decorrelation protocol ensures that the entity is tracked by creating a new track number which is updated by a distinct observer. The former track number is eventually dropped. Naturally, these two protocols change the track-truth link by respectively adding and removing track numbers.

3.1.5 Bounded track error

Because radar systems commonly exhibit imperfections, detections are inherently imperfect. Consequently, observations made by an observer regarding a truth object's position contain errors. The magnitude of these errors can vary across different radar systems, leading to discrepancies in the distances between observed tracks and actual positions. In certain scenarios, accurately characterizing the distribution of these errors can be challenging. However, it is feasible to establish a limit or threshold on the errors. Hence, it is reasonable to assume that the distance between the positions of a track and the corresponding truth is bounded by a predetermined value. This value depends on the observer reporting the track.

3.2 Mathematical description

To formulate the problem of finding a link between track and truth objects, it is necessary to establish mathematical models for both.

Definition 3.1 (Track and Truth)**.** *Let d_1 and d_2 be integers with $d_1 \leq d_2$ and $t \in \mathbb{R}_+$*

(a) *A track is a set $\chi_T = \{t_1, \ldots, t_k\} \subseteq [0, t]$ together with a function $T : \chi_T \longrightarrow \mathbb{R}^{d_1}$.*

(b) *A truth D is a continuous function $D : [0, t] \longrightarrow \mathbb{R}^{d_2}$.*

We will now justify the definition above. Firstly, the dimensions d_1 and d_2 of track and truth objects, dictate the amount of information available about an object. In practice, a track typically contains less information than a truth object, hence $d_1 \leq d_2$. Secondly, as a truth object is simulated over a time interval $[0, t]$ during which accurate information about it is available, it is reasonable to model truth objects as real vector-valued functions over this interval. Since tracks represent truth objects only temporarily, it is natural to model them as sequences of real-valued vectors. From a datalink perspective (Link-16), a mathematical track corresponds to the collection of tracks reported under the same track number. Occasionally, a single vector in a track is also referred to as a track. This should, however, be clear from the context.

Throughout this thesis, the most relevant track and truth information is positional data. Therefore, it is convenient to define a track and truth's position. Positional data can be registered in many ways, though in this thesis we adopt the ECEF cartesian coordinate system (Earth-centered, Earth-fixed). The ECEF coordinate system is obtained by positioning a cartesian coordinate system with its center at the earth's center of mass, then aligning the z-axis with the line between the north and south poles and rotating it along this axis so that the x-axis passes through the prime meridian. As a consequence, the ECEF system contains the equator in the xy-plane. Now, for a track T and time $\tau \in \chi_T$, we denote by $T_\rho(\tau)$ the position of track T at time τ in the ECEF coordinate system. Similarly, for truth D and time $\tau \in [0, t]$ we denote by $D_\rho(\tau)$ the position of truth D at time τ in the ECEF coordinate system.

Before formalizing the notion of a track-truth link, some more preliminaries are required. Firstly, for a set of tracks $\mathcal{T}$ we can distinguish between the observers, of which there are l, that have reported the tracks. Specifically, we may partition the track set $\mathcal{T}$ into sets $\mathcal{T}_1, \ldots, \mathcal{T}_l$, where $\mathcal{T}_i$ consists of the respective tracks reported by observer i. We refer to the partition $\{\mathcal{T}_1, \ldots, \mathcal{T}_l\}$ as the *observer partition*. Furthermore, each observer i has a *confusability range* R_i, that determines the maximum distance between a truth and a track of that truth at a single point in time, reported by observer i. Secondly, we define the *track-truth graph*, which is the complete bipartite graph $G(\mathcal{T}, \mathcal{D}) = (V, E)$ with vertex set $V = \mathcal{T} \cup \mathcal{D}$ and $E = \mathcal{T} \times \mathcal{D}$, having bipartite sets $\mathcal{T}$ and $\mathcal{D}$ respectively. We can now define a track-truth link formally. In the definition, the notation $N_G(v)$ denotes the set of all neighbors of v in the graph G.

Definition 3.2. *Let $\mathcal{T} = \{T_1, \ldots, T_n\}$ be a set of tracks having observer partition $\{\mathcal{T}_1, \ldots, \mathcal{T}_l\}$, and $\mathcal{D} = \{D_1, \ldots, D_m\}$ a set of truths. The track-truth link is a unique function $f : [0, t] \longrightarrow 2^E$ satisfying the following properties, in which $G(\tau) = (V, f(\tau))$*

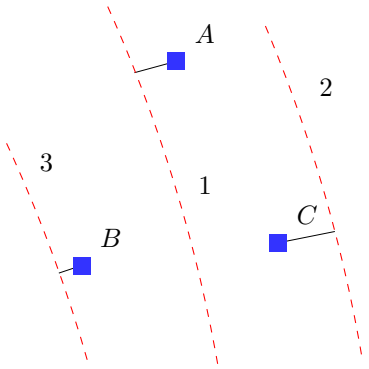
(i) $|N_{G(\tau)}(T_i)| \leq 1$ *for all* $\tau \in [0, t]$ *and* $i \in [n]$.

(ii) $|N_{G(\tau)}(D_j)| \geq 1$ *for all* $\tau \in [0, t]$ *and* $j \in [m]$.

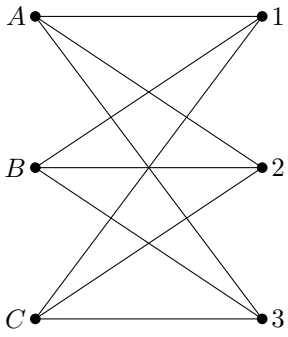
(iii) *Let* $i \in [l]$ *and* $\tau \in [0, t]$, *then* $N_{G(\tau)}(T) \cap N_{G(\tau)}(\widetilde{T}) = \varnothing$ *for distinct* $T, \widetilde{T} \in \mathcal{T}_i$

(iv) *Let* $i \in [l]$, $T \in \mathcal{T}_i$, $\tau \in \chi_T$ *and* $D \in \mathcal{D}$, *then* $(T, D) \in f(\tau) \implies \|T_\rho(\tau) - D_\rho(\tau)\| \leq R_i$.

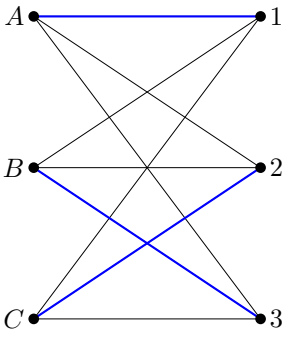
This definition states that the track-truth link is a function that maps each time τ to a set of edges satisfying properties (i) through (iv). Which correspond to the properties described in Sections 3.1.1 through 3.1.5, respectively. Essentially, the edges $f(\tau)$ for a given time $\tau \in [0, t]$ determine the assignment of tracks to truths. An example of a scenario and corresponding track-truth graph and track-truth link can be found in Figure 1.



(a) *Scenario with tracks (blue) and truths (red) whose correspondence is displayed by lines.*



(b) *The track-truth graph corresponding to scenario 1a*



(c) *Correct track-truth link corresponding to scenario 1a*

Figure 1: *Example of scenario with track and truth entities and corresponding track-truth graph and link.*

4 Assignment Problems

4.1 Integral Polyhedra and Total Unimodularity

A fundamental object in combinatorial optimization is an integral polyhedron. Integral polyhedra have the property that all faces contain an integer point. This property is extremely valuable in integer linear programming since there is no need for integral constraints, allowing the usage of linear program solvers. This section explores the concept of total unimodularity, a property of matrices that implies integrality of polyhedra. This section assumes that the reader is familiar with the basics of polyhedra.

We start with the integer hull operation. For a polyhedron P define its *integer hull* P_I as the convex hull of all its integral vectors, i.e.

$$P_I = \text{conv}\{\, x : x \in P,\, x \text{ integral} \,\}$$

A polyhedron P is called *integral* if it is equal to its integer hull, i.e. $P = P_I$. There are many equivalent definitions for integral polyhedra, see [18] for a survey. The crucial property of integral polyhedra is the following result [18].

Theorem 4.1. *Let $P \subseteq \mathbb{R}^n$ be an integral polyhedron and $c \in \mathbb{R}^n$ such that $z = \max\{c^\top x : x \in P\}$ exists and is finite. Then there exists an integral vector $x \in P$ satisfying $c^\top x = z$.*

This result shows that linear programs with integral polyhedra admit optimal integral solutions, though it need not be true that all optimal solutions are integral. Since a polyhedron P is completely determined by linear inequalities $Ax \leq b$, it is only natural to contemplate the existence of a property of A and b that implies integrality of P. Indeed such a result exists and has been known for quite some time. This result relies on the following property.

Definition 4.2. *A matrix A is called totally unimodular if $\det(B) \in \{0, \pm 1\}$ for every square submatrix B of A.*

It follows immediately from the definition that a totally unimodular matrix consists only of entries in $\{0, \pm 1\}$. Total unimodularity is preserved by basic matrix operations, as demonstrated by the next result [18].

Lemma 4.3. *Let A be a totally unimodular matrix and let B be the matrix obtained from A by either:*

(i) *permutating the rows or columns*

(ii) *transposing the matrix*

(iii) *multiplying a row or column by -1*

(iv) *adding a row or column with a single nonzero entry, being ± 1*

(v) *adding a row or column from A*

Then B is totally unimodular as well.

There exist more operations preserving total unimodularity, though for this thesis these are sufficient. Next is a fundamental result due to Alan J. Hoffman and Joseph B. Kruskal [8] that guarantees that a polyhedron is integral.

Theorem 4.4. *Let A be a totally unimodular matrix and b an integral vector, then the polyhedron $P = \{x : Ax \leq b\}$ is integral.*

Given a linear program with constraint matrix A, integral vector b, and polyhedron P induced by the constraints. Ideally, we would like to apply Theorem 4.4 to A to verify whether P is integral. This is possible if the linear program only has lesser-equal constraints, though if it has greater-equal or equality constraints, Theorem 4.4 is not sufficient. Naturally, greater-equal and equality constraints can be enforced via lesser-equal constraints. However, in doing so, the constraint matrix A of the linear program is altered, either by multiplying a row by minus-one for a greater-equal constraint or by repeating a row and multiplying it by minus-one as well (yielding a lesser-equal and greater-equal constraint of the same row).

Fortunately, the property of total unimodularity is preserved by both these operations, as demonstrated by Lemma 4.3. Therefore, if the constraint matrix A of a linear program is totally unimodular, Lemma 4.3 guarantees that there exists a totally unimodular matrix A' such that $P = \{x : A'x \leq b\}$ which is constructed from A by multiplying rows by minus-one and repeating rows. We have essentially demonstrated the following result.

Theorem 4.5. *If the constraint matrix of a linear program is totally unimodular, then the polyhedron induced by these inequalities is integral.*

4.2 Assignment problems

We illustrate the fundamentals of the assignment problem through an example. Suppose you are the owner of a new zoo and wish to assign animals from a set A to exhibits from a set B, under the assumption that no two animals may be assigned to the same exhibit and that the number of exhibits equals the number of animals. Taking into account the welfare of the animals, you want to assign each animal to a suitable exhibit. However, some animals are more suited for an exhibit than others (i.e. a polar bear demands a larger habitat than a frog). To this end, we define a suitability score c_{ab} for each pair ab of animal $a \in A$ and exhibit $b \in B$, which defines how appropriate exhibit b is for animal a. Exhibits that greatly mimic the habitat of animals are given a large suitability score, and vice versa. A suitable choice for an assignment that maximizes the total welfare of the animals is to choose the set of pairs ab that maximize the sum of suitability scores.

There are many variations of the assignment problem, four of which will be of main interest to us. Firstly, the assignment problem sketched above is more explicitly referred to as the *balanced assignment problem*, indicating that the sets of animals and exhibits are of equal cardinality (so balanced). Secondly, there is the *unbalanced assignment problem* (UAP for brevity) which is identical to the balanced assignment problem, only that it allows for sets of different cardinality. In our example, this corresponds to assigning animals from A to a larger number of exhibits B. Thirdly, there is the *bottleneck assignment problem* (BAP for brevity). The bottleneck assignment problem has identical constraints as the unbalanced assignment problem, the only difference lies in the objective. Rather than minimizing the sum of suitability scores, we minimize the maximum suitability score (or maximize the minimum value) of our assignment. Again, in our example, this means we want to find an assignment of animals to exhibits such that the smallest suitability score (in our assignment) is maximized. Note that because the bottleneck assignment problem has identical constraints to the unbalanced assignment problem, it also accounts for sets of different cardinality. Lastly, *lexicographic bottleneck assignment* aims to find a lexicographic minimal solution to the bottleneck assignment. It may be thought of as a secondary optimization among all feasible solutions to the bottleneck assignment problem.

4.2.1 Balanced assignment problem

We now formulate this problem concretely in a graph-theoretical sense. Consider a complete bipartite graph $G = (V, E)$ with bipartite sets $V = V_1 \cup V_2$ with $|V_1| = |V_2| = n$. A *matching* in G is a set $M \subseteq E$ of edges such that no two edges in M share an endpoint. A matching M in G is of *maximum cardinality* if there does not exist a matching of larger cardinality. Let $C \in \mathbb{R}^{n \times n}$ be the *cost matrix* that associates with each edge ij a cost C_{ij}. The objective of the balanced assignment problem is to find a maximum cardinality matching in G that minimizes (or maximizes) the sum of costs of edges in M, i.e.

$$\text{argmin}\Big\{ \sum_{e \in M} c_e \ : \ M \text{ is a maximum cardinality matching of } G \Big\} \tag{1}$$

Depending on the context and nature of the problem, we may write this as a $\min$ problem if the exact solution is not of importance. The assignment problem may be extended to an incomplete bipartite graph G by assigning sufficiently large (or small) costs to the non-existing edges in G. Sufficiently large, in this sense, means large enough so that they will not be part of an optimal solution.

We can write the balanced assignment problem as an integer program. With each maximum cardinality matching M in G we associate a binary matrix $X \in \mathbb{R}^{n \times n}$ by setting $X_{ij} = 1$ if $ij \in M$ and zero otherwise. The matrix X may be thought of as an adjacency matrix for bipartite graphs sometimes referred to as the *biadjacency matrix*. Since M is a maximum cardinality matching, all row and column sums evaluate to 1 exactly, indicating that X is a permutation matrix. On the other hand, every permutation matrix $X \in \mathbb{R}^{n \times n}$ corresponds to a maximum cardinality matching in G by selecting the edges $ij \in E$ such that $X_{ij} = 1$. Hence, we may characterize maximum cardinality matchings with permutation matrices, leading to the following integer programming formulation

$$\begin{aligned} \min_{X \in \mathbb{R}^{n \times n}} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{n} X_{ij} = 1 \qquad i = 1, \ldots, n \\ & \sum_{i=1}^{n} X_{ij} = 1 \qquad j = 1, \ldots, n \\ & X_{ij} \in \{0, 1\} \qquad i, j = 1, \ldots, n \end{aligned} \tag{2}$$

The appearance of permutations in this context is no coincidence as is demonstrated by the next line of thought in our example. Rather than assigning n animals to n exhibits, we may view the problem as assigning each animal uniquely to a number in $\{1, \ldots, n\}$, ordering them. In this context, we optimize over all possible orders of the numbers $\{1, \ldots, n\}$, rather than the explicit assignments. Since the orderings of these numbers are identically described by the permutations of size n, we may optimize over all permutations of this size.

As verified in [3], the binary constraint in (2) may be relaxed to $X_{ij} \geq 0$ due to total unimodularity. Recall from Theorem 4.4 that polyhedra which are induced by a totally unimodular matrix and integral vector are in fact integral. The constraint matrix of (2) is

$$A_{\text{assign}} = \left(\begin{array}{cccccccccccccccc}
1 & 1 & \cdots & 1 & & & & & & & & & & & & \\
 & & & & 1 & 1 & \cdots & 1 & & & & & & & & \\
 & & & & & & & & 1 & 1 & \cdots & 1 & & & & \\
 & & & & & & & & & & & & 1 & 1 & \cdots & 1 \\
\hline
1 & & & & 1 & & & & 1 & & & & 1 & & & \\
 & 1 & & & & 1 & & & & 1 & & & & 1 & & \\
 & & \ddots & & & & \ddots & & & & \ddots & & & & \ddots & \\
 & & & 1 & & & & 1 & & & & 1 & & & & 1
\end{array}\right)$$

It was already verified in [3] that A_{assign} is totally unimodular. However, their result relied on other results and for that reason, we supply a self-contained proof.

Lemma 4.6. *The matrix A_{assign} is totally unimodular.*

Proof. Let A_{assign} be of dimension n and let B be a square submatrix of A_{assign}. We prove the statement via the definition of total unimodularity and induction on the dimension of B. For the base case, assume that B consists of a single row and column. Then $B = (1)$, and therefore $\det(B) = 1$. Proceeding with the induction hypothesis, suppose B has dimension $k \geq 2$ and that $\det(B') \in \{0, \pm 1\}$ for all square submatrices B' of A_{assign} with dimension strictly smaller than k. It is easily verified from A_{assign} that B must have a column that contains at most one nonzero entry. If B has the zero vector as a column, then clearly $\det(B) = 0$. Contrarily, if B does not contain the zero vector as a column, then it has a column with exactly one 1. Now the determinant of B may be expanded over this column, hence it may be expressed as $\det(B) = \pm \det(B')$, where B' is a square submatrix of A_{assign} consisting of $k-1$ rows and columns. By the induction hypothesis, we have that $\det(B') \in \{0, \pm 1\}$, implying that $\det(B) \in \{0, \pm 1\}$ as well. Therefore, the matrix A_{assign} is totally unimodular. □

Due to Lemma 4.6 and Theorem 4.5, the linear program obtained from (2) by removing the binary constraint has an integer optimal solution. Furthermore, if the binary constraints are replaced with constraints $X_{ij} \geq 0$, the row and column-sum constraints would guarantee that $X_{ij} \leq 1$ as well. Due to Lemma 4.3, replacing binary constraints with non-negativity constraints will preserve total unimodularity. Therefore, the linear program with the replaced constraints and integer program (2), share an integral optimal solution.

4.2.2 Unbalanced assignment problem

The *unbalanced assignment problem* (or UAP for the sake of brevity) is practically identical to the regular assignment problem, except for the cardinalities of the bipartite set of G, which may take arbitrary sizes, say $|V_1| = n$ and $|V_2| = m$ with $n \leq m$. The objective of the UAP is to find a maximum cardinality matching in G that minimizes the total cost, so identical to that of the balanced assignment problem. The integer programming formulation (2) for the balanced assignment problem may easily be extended to a formulation of the unbalanced assignment problem by varying the dimensions of matrix X

$$\begin{aligned} \min_{X \in \mathbb{R}^{n \times m}} \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij} X_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{m} X_{ij} = 1 \qquad i = 1, \ldots, n \\ & \sum_{i=1}^{n} X_{ij} \leq 1 \qquad j = 1, \ldots, m \\ & X_{ij} \in \{0, 1\} \qquad i = 1, \ldots, n \quad j = 1, \ldots, m \end{aligned} \tag{3}$$

Since the constraint matrix of (3) is identical to that of (2), the binary constraint of the unbalanced assignment problem may also be relaxed to $X_{ij} \geq 0$.

In addition to (3), the dual formulation of the unbalanced assignment problem will be of importance. Using basic duality theory, the dual of (3) with relaxation $X_{ij} \geq 0$ can be derived. This is

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} u_i + \sum_{j=1}^{m} v_j \\ \text{s.t.} \quad & u_i + v_j \leq C_{ij} \qquad i = 1, \ldots, n \quad j = 1, \ldots, m \\ & v_j \leq 0 \qquad\qquad\quad j = 1, \ldots, m \end{aligned} \tag{4}$$

4.2.3 Bottleneck assignment problem

The third variant of the assignment problem that will be of interest is the *bottleneck assignment problem* (or BAP). The term *bottleneck* refers to the cost that prevents an increase (or decrease) in the optimal value of the problem. The constraints of a bottleneck assignment problem are identical to those of the unbalanced assignment problem. The objective function, however, is the maximal cost of an edge within the matching. Changing the objective function in formulation (3) leads to the following formulation for

the bottleneck assignment problem

$$\begin{aligned} \min_{X \in \mathbb{R}^{n \times m}} \quad & \max_{i \in [n], j \in [m]} C_{ij} X_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{m} X_{ij} = 1 \qquad i = 1, \ldots, n \\ & \sum_{i=1}^{n} X_{ij} \leq 1 \qquad j = 1, \ldots, m \\ & X_{ij} \in \{0, 1\} \qquad i = 1, \ldots, n \quad j = 1, \ldots, m \end{aligned} \tag{5}$$

Formulation (5) of the bottleneck assignment problem is not linear due to the objective function. However, this may be achieved by introducing an artificial variable z, adding the constraints $C_{ij}X_{ij} \leq z$ for $i \in [n]$, $j \in [m]$ and replacing the current objective function by z. In this case, the property of total unimodularity of the constraint matrix is lost due to the newly added constraints with real coefficients C_{ij} for X_{ij}.

Inspection of the objective function shows that the objective value of (5) will be a cost coefficient C_{ij} for some $ij \in E$. Therefore, the optimal value of the bottleneck assignment problem will be one of the entries in the cost matrix C. In the balanced case, i.e. $n = m$, this gives rise to the following elegant formulation for the bottleneck assignment problem [3]

$$\min_{\varphi \in S_n} \max_{1 \leq i \leq n} C_{i\varphi(i)}.$$

Here S_n denotes the set of permutations of size n, i.e. the symmetric group.

4.2.4 Lexicographic Bottleneck Assignment problem

Lastly, a variation of the bottleneck assignment problem is discussed, called *lexicographic bottleneck assignment* or LexBAP for the sake of brevity. In order to understand LexBAP the notion of the *lexicographic order* is required. The lexicographic order for $x, y \in \mathbb{R}^n$ is defined as

$$x \prec y \iff \exists k \in [n] \, : x_k < y_k \wedge x_i = y_i \; \forall i \leq k - 1.$$

Intuitively, ordering as set according to the lexicographic order is identical to sorting a dictionary in the natural way (alphabetically). This is expected as the lexicographic order is inspired by the alphabetical order of dictionaries. Furthermore, the lexicographic order defines a total order on $\mathbb{R}^n$.

The objective of LexBAP is to find a solution to the bottleneck assignment problem that has a lexicographic minimal cost vector. Lexicographic bottleneck may be thought of as a secondary optimization among all optimal solutions to the bottleneck assignment problem. At present, no integer programming formulations for LexBAP are known. We do give a more formal description here. For a feasible solution

X of (5) (BAP) define $c(X)$ to be a vector consisting of entries C_{ij} for which $X_{ij} = 1$ that is sorted decreasingly. The formulation below now describes lexicographic bottleneck assignment, in which the subscript $\preceq$ suggests that the objective is to minimize according to the lexicographic order.

$$\begin{aligned} \min_{\preceq,\, X \in \mathbb{R}^{n\times m}} \quad & c(X) \\ \text{s.t.} \quad & \sum_{j=1}^{m} X_{ij} = 1 \qquad i = 1, \dots, n \\ & \sum_{i=1}^{n} X_{ij} \leq 1 \qquad j = 1, \dots, m \\ & X_{ij} \in \{0, 1\} \qquad i = 1, \dots, n \quad j = 1, \dots, m \end{aligned} \tag{6}$$

4.3 Augmenting paths and the Hopcroft-Karp-Karzanov algorithm

4.3.1 Augmenting paths and Berge's theorem

An important notion in the solution methods for assignment problems is that of an *augmenting path*. It was (probably) first introduced by Berge in [2] to derive a criterion for characterizing whether a matching is of maximum cardinality. Prior to the definition of an augmenting path, we require the notion of a free vertex. For a matching M of a graph $G = (V, E)$ and a vertex $v \in V$ we say v is *free* with respect to M if it is not incident to an edge in M.

Definition 4.7. *Let M be a matching in graph $G = (V, E)$. An M-augmenting path in G is a path $P = v_1\, v_2 \dots v_k$ such that v_1 and v_k are free with respect to M and the edges are alternatively in M and $E - M$.*

The notion introduced by Berge in [2], called an *alternating chain*, only required that for any two adjacent edges in the path P, one belongs to M and the other to $E - M$ and not that the start- and endpoints of the chain are free with respect to M. Augmenting paths allow the construction of matchings of larger cardinality, as demonstrated by the next lemma.

Lemma 4.8. *Let M be a matching and P and M-augmenting path. Then $M' = M \oplus P$ is a matching as well and $|M'| = |M| + 1$.*

Proof. Using the definition of the symmetric difference we have $M' = (M \setminus P) \cup (P \setminus M)$, which is the union of two disjoint sets. Since the sets $P \setminus M$ and $M \setminus P$ are both matchings, it is sufficient to verify that the edges in these sets do not share a vertex. Suppose such a vertex does exist, then it must be either the startpoint, endpoint, or an internal vertex (not the start- or endpoint) of P. It cannot be an internal vertex of P, as all such vertices are adjacent to some edge in $M \cap P$, which would contradict that M is a matching. Additionally, it cannot be the start or endpoint of P as they are free with respect to M. Therefore such vertices do not exist, hence M' is a matching. Now consider the cardinality of M'.

Assume that $|M \setminus P| = x$, then $|P \setminus M| = |M| - x + 1$ as P is M-augmenting and $P \setminus M \subseteq P$. Together this gives

$$|M'| = x + |M| - x + 1 = |M| + 1.$$

□

Next is Berge's theorem [2], which is an important result regarding augmenting paths, characterizing whether a matching is of maximum cardinality. Even though Petersen [17] derived a closely related result before, it is named after Berge. See [12] for a more detailed description.

Theorem 4.9 (Berge)**.** *A matching M in G is of maximum cardinality if and only if there does not exist an M-augmenting path.*

Proof. We argue via contraposition, so assume there exists an M-augmenting path P. Then $M' = M \oplus P$ is a matching of larger cardinality. Hence M is not a maximum cardinality matching.
Now assume M is not of maximum cardinality. Thus there exists a matching M' of larger cardinality than M. Now consider the subgraph G' of G induced by the edges $M \oplus M'$. Since M and M' are matchings, every vertex in G' has degree 1 or 2. The connected components of G' are exactly cycles of even length and paths (of arbitrary length) whose edges belong alternatively to M and M'. Since $|M'| > |M|$ there must exist a path P of odd length that has more edges in M' than in M. This must be an M-augmenting path, concluding the proof. □

Definition 4.10. *For sets A and B the symmetric difference $A \oplus B$ is defined as*

$$A \oplus B = (A \setminus B) \cup (B \setminus A)$$

The symmetric difference operation is commutative and associative, so that we may easily write the symmetric difference of multiple sets $A_1, \ldots, A_k$ as $A_1 \oplus A_2 \oplus \cdots \oplus A_k$, independent of the order. For augmenting paths, the symmetric difference may be thought of as choosing the edges in M that are not P and the edges in P that are not in M.

4.3.2 The Hopcroft-Karp-Karzanov algorithm

The renowned Hopcroft-Karp-Karzanov algorithm or HKK-algorithm for brevity, finds a maximum cardinality matching in an unbalanced bipartite graph G. It was independently discovered by John. E. Hopcroft, Richard M. Karp [9] and Alexander V. Karzanov [11]. It has, for a long time, been the maximum cardinality matching algorithm with the best worst-case scenario performance of $\mathcal{O}(m\sqrt{n})$ for a graph on n vertices and m edges. This section explains the HKK-algorithm, following the description of Hopcroft and Karp [9].

The HKK-algorithm computes a maximum cardinality matching by iteratively constructing a set $\mathcal{P}_M = \{P_1, \ldots, P_l\}$ of M-augmenting paths such that: each path in $\mathcal{P}_M$ is of minimal length (among all the M-augmenting paths); the paths are vertex disjoint; the set is maximal (in the sense that it is not contained in another set with the same properties). It then constructs a new matching $\tilde{M}$ of larger cardinality by computing the symmetric difference with M over the paths in $\mathcal{P}_M$ i.e

$$\tilde{M} = M \oplus P_1 \oplus \cdots \oplus P_l$$

This indeed gives a matching as the augmenting paths in $\mathcal{P}_M$ are vertex disjoint. In the next iteration, we use $M = \tilde{M}$. The algorithm terminates when there no longer exist M-augmenting paths, hence Berge's theorem guarantees that the matching M found by the algorithm is of maximum cardinality. The exact procedure of the Hopcroft-Karp-Karzanov algorithm is described in Algorithm 1.

Algorithm 1 provides a straightforward method to compute a maximum cardinality matching. It even computes such a matching for an arbitrary graph. However, the algorithm does not yet specify how to compute the set of vertex disjoint shortest M-augmenting paths in each iteration. The implementation by Hopcroft and Karp 1973 achieves this for a bipartite graph G by constructing a new graph $\hat{G}$ in which vertex disjoint paths between free vertices correspond to vertex disjoint shortest M-augmenting paths in the original graph. It then finds these paths using a depth-first search.

We consider the construction of the graph $\hat{G}$. Let $G = (V, E)$ be bipartite with $V = V_1 \cup V_2$ with matching M. Firstly, we consider a directed copy $\bar{G} = (V, \bar{E})$ of G such that edges in M are oriented from V_1 to V_2 and directions flipped for edges not in M. We define $F_1 \subseteq V_1$ and $F_2 \subseteq V_2$ to be the sets of free vertices with respect to M. Initialize $L_0 = F_1$ and iteratively construct the sets L_i of vertices and

Algorithm 1 Hopcroft-Karp-Karzanov

$M \leftarrow \varnothing$
while $P \neq \varnothing$ **do**
  $P \leftarrow$ ConstructMaximalPaths(G, M)
  $M \leftarrow M \oplus P_1 \oplus P_2 \oplus \cdots \oplus P_k$
return M

procedure ConstructMaximalPaths(G, M)
  Construct a maximal set $P = \{P_1, P_2, \ldots, P_k\}$ such that
    (a) the P_i are shortest M-augmenting paths in G;
    (b) the P_i are vertex disjoint.
  return P

E_i of directed edges as

$$E_i = \{(u,v) \,:\, uv \in \bar{E},\ v \in L_i,\ u \notin L_0 \cup \cdots \cup L_i\}$$
$$L_i = \{u \,:\, (u,v) \in E_i \text{ for some } v\}$$

The sets L_i and E_i will respectively construct the vertex en edge sets of the graph $\hat{G}$. Now let $\Delta = \min\{i \,:\, L_i \cap F_2 \neq \varnothing\}$ and construct $\hat{G} = (\hat{V}, \hat{E})$ with

$$\hat{V} = L_0 \cup L_1 \cup \cdots \cup L_{\Delta-1} \cup (L_\Delta \cap F_2)$$
$$\hat{E} = E_0 \cup E_1 \cup \cdots \cup E_{\Delta-2} \cup \{(u,v) : v \in L_{\Delta-1},\ u \in F_2\}$$

Intuitively, the directed graph $\hat{G}$ can be visualized with layers consisting of vertices L_0 through $L_\Delta \cap F_2$, with edges between these layers according to E_0 through $\{(u,v) : v \in L_{\Delta-1},\ u \in F_2\}$. The parameter Δ equals the length of a shortest M-augmenting path in G, which explains the sets $L_\Delta \cap F_2$ rather than L_Δ and $\{(u,v) : v \in L_{\Delta-1},\ u \in F_2\}$ rather than $E_{\Delta-1}$, because only M-augmenting paths of this length are considered. Figure 2 gives an example of a graph G and the construction of $\hat{G}$. We now verify the claimed properties of the graph $\hat{G}$.

Theorem 4.11. *Let $G = (V,E)$ be a bipartite graph and $\hat{G}$ defined as above. Let $\hat{Q}$ be a maximal set of vertex disjoint directed paths in $\hat{G}$ starting at a vertex in F_2 and ending at a vertex in F_1. Then the set Q consisting of the undirected variants of paths in $\hat{Q}$ is a maximal set of vertex disjoint shortest M-augmenting paths in G.*

Proof. We first show that each directed path $\hat{P}$ in $\hat{G}$ that starts in a vertex of F_2 and ends at a vertex in F_1, corresponds to a shortest M-augmenting path in G. Let P be the undirected variant of $\hat{P}$, then P is a path in G. Furthermore, the start and endpoint of P are free with respect to M as they belong to F_2 and F_1 respectively. Additionally, because the edges in the directed copy $\bar{G}$ are oriented according to M, the edges of $\hat{P}$ must belong alternatively to M and $E - M$. Therefore, P is an M-augmenting path in G. Next, we show that P is in fact a shortest M-augmenting path. Note firstly that vertices from F_2 appear only in the vertex set $L_\Delta \cap F_2$ from $\hat{G}$, and that vertices from F_1 appear only in L_0. Since $\hat{P}$ starts

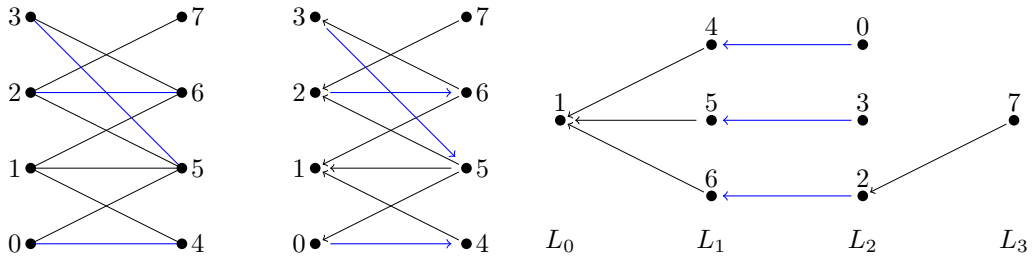


Figure 2: *Example of graphs in HKK-algorithm. Left: Example graph G. Center: The converted graph $\bar{G}$. Right: The final graph $\hat{G}$.*

and ends with vertices from these sets, the length of path $\hat{P}$ (and P) is Δ. Now suppose there exists an M-augmenting path $\tilde{P}$ in G with length $k < \Delta$. Then the directed variant of $\tilde{P}$ is a directed path in $\hat{G}$, implying either that a vertex of F_2 belongs to $L_{\Delta-k}$ or that a vertex from F_1 belongs to L_k (or both). This is a contradiction, as such vertices appear only in $L_\Delta \cap F_2$ and L_0 respectively. Hence P is a shortest M-augmenting path in G.
For the statement, it is sufficient that vertex disjoint directed paths in $\hat{G}$ correspond to vertex disjoint paths in G. Since the vertices of the paths $\hat{P}$ and P are identical, this is trivial and we are done. □

A large portion of the paper [9] is dedicated to the determination of the complexity of Algorithm 1. The most important result, regarding complexity, is that Algorithm 1 terminates within $2\lfloor\nu(G)\rfloor + 2$ iterations, where $\nu(G)$ is the matching number of the graph G [9]. The construction of $\hat{G}$ and the depth-first search for finding a maximal set of vertex disjoint paths in $\hat{G}$ may be achieved in $\mathcal{O}(n+m)$ iterations. Since $\nu(G) = \mathcal{O}(n)$ and $m = \mathcal{O}(n^2)$, the complexity of Algorithm 1 is $\mathcal{O}(n^{2\frac{1}{2}})$. In [1], the method of Hopcroft and Karp was improved to a complexity of $\mathcal{O}(n^{1.5}/\sqrt{m\log(n)})$ by using clever data structures and an adjacency matrix scanning technique due to [4]. Their method provides an improvement of factor $1/\sqrt{\log n}$ for dense graphs, where dense is understood as $m = \Theta(n^2)$.

In [11], Karzanov formulated the procedure as a maximal flow problem. Consider a directed variant G_{dir} of G obtained by orienting the edges from V_1 to V_2, and adding a *source* s incident to the vertices in V_1 (oriented s to V_1) and *sink* t incident to V_2 (oriented V_2 to t). Additionally, assign each edge unit weight. Karzanov observed that finding a maximum cardinality matching in a bipartite graph, is equivalent to finding a maximum st-flow in G_{dir}. By applying a maximum flow algorithm [6] to G_{dir}, Karzanov obtained an algorithm with identical complexity.

4.4 Threshold algorithm for bottleneck assignment

In this section a threshold algorithm [3] solving the bottleneck assignment problem is discussed. The algorithm iteratively lowers an upper bound on the optimal value, called the threshold, until there no longer exist matchings of maximum cardinality.

Recall the bottleneck assignment problem stated in Section 4.2.3, whose formulation is repeated here for the sake of convenience.

$$\begin{aligned} \min_{X \in \mathbb{R}^{n \times m}} \quad & \max_{i \in [n], j \in [m]} C_{ij} X_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{m} X_{ij} = 1 \qquad i = 1, \ldots, n \\ & \sum_{i=1}^{n} X_{ij} \leq 1 \qquad j = 1, \ldots, m \\ & X_{ij} \geq 0 \qquad\quad i = 1, \ldots, n \quad j = 1, \ldots, m \end{aligned} \tag{5}$$

The objective of the bottleneck assignment problem is to find a maximum cardinality matching M in the bipartite graph $G = (V, E)$ (with $V = V_1 \cup V_2$, $|V_1| = n$, $|V_2| = m$) that minimizes the maximum edge cost in the matching, which is dictated by the cost matrix $C \in \mathbb{R}^{n \times m}$. We call the optimum value $z^\star$ of (5) the *bottleneck value* for the bottleneck assignment problem with cost matrix C.

A simple yet effective method for solving (5) is to repeatedly decrease an upper bound or *threshold* on the entries of C until the graph consisting of edges with cost lower than this threshold no longer contains a maximum cardinality matching. The smallest threshold for which this graph still contains a maximum cardinality matching must be the bottleneck value $z^\star$.

Let us formalize this method into an algorithm. At the start of the algorithm an interval $[c_0, c_1]$ is initialized so that $z^\star \in [c_0, c_1]$. The value c_1 is the threshold value mentioned priorly. Subsequently, a value $c \in C^\star = \{C_{ij} : c_0 < C_{ij} < c_1\}$ is chosen as a contender for $z^\star$. For this c the *threshold graph* $G[c]$ is defined as the graph with vertices V and edges $E(G) = \{ij \in E : C_{ij} \leq c\}$. There are now two cases possible, either $G[c]$ has a maximum cardinality matching or it does not. In the first case, it must be that $c \geq z^\star$. On the other hand, if $G[c]$ does not contain a maximum cardinality matching, it must be that $c \leq z^\star$. In the next iteration, our search space can be narrowed down by either decreasing c_1 to c or increasing c_0 to c, depending on whether $G[c]$ has a maximum cardinality matching or not. This process repeats until the set $C^\star$ is empty. At this stage, we have either $z^\star = c_0$ or $z^\star = c_1$. If $G[c_0]$ contains a maximum cardinality matching, then it follows that $z^\star = c_0$. Contrarily, if $G[c_0]$ does not contain a maximum cardinality matching, then $z^\star = c_1$ as $G[c_1]$ does. In order to prevent checking $G[c_0]$ for existence of a maximum cardinality matching, we can keep a set C_{checked} of checked values[1].

[1] Note that the only case for which $c_0 \notin C_{\text{checked}}$ is when c_0 equals the initialized value.

Let us address some minor details of the threshold algorithm. Firstly, in the initialization of c_0 and c_1 it is required that $z^\star \in [c_0, c_1]$. When no additional information about the bottleneck value is available, the customary choice is to take the minimal cost and maximal cost of the cost matrix for c_0 and c_1. Secondly, it is not yet specified how the algorithm chooses a contender $c \in C^\star$. For this a binary search approach is adopted, i.e. selecting the median of $C^\star$ for the contender c in the current iteration. A pseudo-code version of the threshold algorithm may be found in Algorithm 2. A detailed example is discussed in [3].

Algorithm 2 Threshold

$c_0 \leftarrow \min C_{ij},\ c_1 \leftarrow \max C_{ij}$
$C^\star \leftarrow \{C_{ij} : c_0 < C_{ij} < c_1\}$
$C_{\text{checked}} \leftarrow \varnothing$

while $C^\star \neq \varnothing$ **do**
    $c \leftarrow$ **median**$(C^\star)$
    if ContainsMCM(c) **then**
        $c_1 \leftarrow c$
    else
        $c_0 \leftarrow c$
    $C_{\text{checked}} \leftarrow C_{\text{checked}} \cup \{c\}$

if $c_0 \in C_{\text{checked}}$ **then**
    return c_1
else
    if ContainsMCM(c_0) **then**
        return c_1
    else
        return c_0

procedure ContainsMCM(c)
    if $G[c]$ contains a maximum cardinality matching **then**
        return True
    else
        return False

We now turn our attention to the complexity of the threshold algorithm using binary search for a cost

matrix $C \in \mathbb{R}^{n \times m}$, with $n \geq m$. Recall that the complexity of binary search for a set of cardinality nm is $O(\log nm)$, which reduces to $O(\log n)$ under the assumption $n \geq m$. Given a procedure of complexity $T(n)$ that checks whether a graph contains a maximum cardinality matching, the overall complexity of the threshold algorithm is $O(T(n) \log n)$. If the Hopcroft-Karp-Karzanov algorithm is used for finding a maximum cardinality matching, the threshold algorithm yields a complexity of $O(n^{2\frac{1}{2}} \log n)$. Note that the complexity remains unchanged in terms of $n + m$, the total number of vertices.

4.5 A solution method for lexicographic bottleneck assignment

In this section, we discuss a solution method from [19] for the lexicographic bottleneck assignment problem (LexBAP) introduced in Section 4.2.4. Recall the objective of LexBAP: find a feasible solution to the bottleneck assignment problem with a lexicographic minimal cost vector. In [19] a solution method for LexBAP is derived for the balanced case. However, due to their method's dependency on the linear programming formulation of the balanced assignment problem and the differences between the formulations for the unbalanced and balanced assignment problem, the method cannot be applied straight away to the unbalanced case. Therefore, we adapt the method in [19] so that it may be used in the unbalanced case.

We first give an intuitive description of the method. Considering that solutions to LexBAP(C) are solutions to BAP(C) as well, an elementary idea is to order the optimal solutions (cost vectors) of BAP(C) according to the lexicographic order and then pick the smallest one. This method seems favorable, however, it does not work well in practice as it is difficult to determine the set of optimal BAP(C) solutions explicitly. Thus a different approach is considered. Suppose the instance BAP(C) has bottleneck value $z = z^\star$. Then the value z must appear a minimal number of times in the cost vector associated with the optimal solution $X^\star$ of LexBAP(C) (it appears at least once). The same must hold for the second largest entry of C after z and the third largest etc. By iteratively performing a simple minimization procedure we can select the solutions that use each cost entry of C smaller than z a minimal number of times, i.e. the optimal solutions of LexBAP(C).

Let us proceed in more detail. Firstly, define the matrix $D = D[z] \in \mathbb{R}^{n \times m}$ as

$$D[z]_{ij} = \begin{cases} 1 & \text{if } C_{ij} = z \\ 0 & \text{else} \end{cases}$$

In order to find solutions that use z a minimal number of times in the cost vector, we can solve an unbalanced assignment problem (or balanced in the case $n = m$) with cost matrix D defined above, together with some additional constraints in order to avoid selecting indices with a value larger than z. Concretely, the constraints $X_{ij} = 0$ are added for every index $(i, j) \in \mathcal{I}$, where $\mathcal{I}$ is the set of indices for which $C_{ij} > z$. For future reasons, the constraints $\sum_{i=1}^{n} X_{ij} = 1$ are added for $j \in \mathcal{J}$, where for now $\mathcal{J} = \varnothing$.

The exact formulation of this problem is

$$\begin{aligned}
\min_{X \in \mathbb{R}^{n\times m}} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} D_{ij}X_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{m} X_{ij} = 1 && i = 1,\dots,n \\
& \sum_{i=1}^{n} X_{ij} \leq 1 && j \notin \mathcal{J} \\
& \sum_{i=1}^{n} X_{ij} = 1 && j \in \mathcal{J} \\
& X_{ij} \geq 0 && (i,j) \notin \mathcal{I} \\
& X_{ij} = 0 && (i,j) \in \mathcal{I}
\end{aligned} \tag{7}$$

From hereon problem (7) will be referred to as UAP$(D, \mathcal{I}, \mathcal{J})$ where $D, \mathcal{I}, \mathcal{J}$ are respectively the cost matrix, index set for constraints $X_{ij} = 0$ and the column index set for constraints $\sum_{i=1}^{m} X_{ij} = 1$. By inspection, it is clear that an optimal solution of (7) contains the cost entry z a minimal number of times in the associated cost vector. Therefore, the set of optimal solutions to (7) contains all the optimal solutions to LexBAP(C), in addition to other solutions. Similar to the issue with BAP, the set of optimal solutions of (7) is difficult to characterize. However, duality combined with complementary slackness in linear programming, can be utilized to characterize the optimal solutions. The dual formulation of (7) is required for this and reads

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} u_i + \sum_{j=1}^{m} v_j \\
\text{s.t.} \quad & u_i + v_j \leq D_{ij} && (i,j) \notin \mathcal{I} \\
& v_j \leq 0 && j \notin \mathcal{J}
\end{aligned} \tag{8}$$

We hereon refer to problem (8) as DualUAP$(D, \mathcal{I}, \mathcal{J})$. For primal (7) and its dual (8), the complementary slackness conditions are as follows.

Theorem 4.12 (Complementary slackness)**.** *Let X and (u, v) be feasible solutions to* UAP$(D, \mathcal{I}, \mathcal{J})$ *and* DualUAP$(D, \mathcal{I}, \mathcal{J})$ *respectively. Then X and (u, v) are both optimal if and only if*

$$\begin{aligned}
v_j \left(\sum_{i=1}^{n} X_{ij} - 1 \right) &= 0, \quad j \notin \mathcal{J} \\
X_{ij} \left(D_{ij} - u_i - v_j \right) &= 0, \quad (i,j) \notin \mathcal{I}
\end{aligned}$$

These complementary slackness conditions can be used to derive constraints for optimal solutions of the primal problem, given an optimal solution to the dual problem. It is easily derived that the complementary

slackness conditions are equivalent to

$$\begin{aligned} v_j < 0 \ \wedge \ j \notin \mathcal{J} \implies \sum_{i=1}^{n} X_{ij} = 1 \\ D_{ij} > u_i + v_j \implies X_{ij} = 0 \end{aligned} \tag{9}$$

In other words, for an optimal dual solution (u, v) we can derive what conditions optimal primal solutions must satisfy. Any feasible primal solution that satisfies the constraints implied in (9), must be optimal. Every index (i, j) for which $v_j < 0$ and $D_{ij} > u_i + v_j$ is added to the set $\mathcal{I}$ and the column index j is added to $\mathcal{J}$. These indices prescribe which constraints to add in the subsequent iterations in order to restrict to optimal solutions.

In the next iteration, we repeat a similar procedure. First, a new value z_{new} is determined by one of two methods. Either by selecting $z_{\text{new}} = \max\{C_{ij} \ : \ C_{ij} < z\}$ (the second largest entry of C after z) or by solving an instance of BAP with cost matrix

$$\widetilde{C}[z] = \begin{cases} \infty & C_{ij} > z \\ 0 & C_{ij} = z \\ C_{ij} & \text{else} \end{cases}$$

The bottleneck value of this instance will be the new value of z_{new}. In the context of the original bottleneck assignment problem, the value of z_{new} is the smallest second largest value among all cost vectors of optimal BAP solutions. If the second method is adopted, then the indices (i, j) for which $z > C_{ij} > z_{\text{new}}$ are added to $\mathcal{I}$. Then, problem (7) is formulated and its dual is solved. This procedure repeats until $z = z_{\text{min}}$. For the smallest z in the cost matrix C, there is no need to solve the dual. Instead, it is more efficient to solve the primal, since at this stage every optimal solution to (7) is an optimal solution to LexBAP(C).

Interestingly, we can conclude that the solution method in [19] for LexBAP in the balanced case may also be applied to the unbalanced case. This new extension differs from the original in the complementary slackness conditions, leading to the addition of new constraints in the dual formulation (column sum constraints).

Algorithm 3 LexBAP

$z \leftarrow \max\{C_{ij} : i \in [n],\, j \in [m]\}$
$z_{\min} = \min\{C_{ij} : i \in [n],\, j \in [m]\}$
$\mathcal{I} \leftarrow \varnothing,\ \mathcal{J} \leftarrow \varnothing$

while $z \geq z_{\min}$ **do**
  $D \leftarrow D[z]$
  $(u, v) \leftarrow$ DualUAP$(D, \mathcal{I}, \mathcal{J})$
  for $i \in [n],\, j \in [m]$ **do**
    if $D_{ij} > u_i + v_j$ **and** $v[j] < 0$ **then**
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{(i, j)\}$
      $\mathcal{J} \leftarrow \mathcal{J} \cup \{j\}$
  $z \leftarrow \max\{C_{ij} : C_{ij} < z\}$
$D \leftarrow D[0]$
$X^\star \leftarrow$ UAP$(D, \mathcal{I}, \mathcal{J})$
return $X^\star$

4.6 Ranking solutions of the unbalanced assignment problem

In applications of the unbalanced assignment problem (UAP), it is worthwhile not just to determine the optimal solution, but also the ones following it in terms of objective value. Essentially, we wish for a method that can determine for every integer k a sequence $X_1, \ldots, X_k$ of solutions to UAP such that $X_1 = X^*$ (optimal solution) and X_i is the next best solution (in terms of objective) after removing $X_1, \ldots, X_{i-1}$. This section presents a method that ranks an arbitrary number of solutions to the unbalanced assignment problem, from optimal to less optimal. The method in [13] is only considered for the balanced case, though it can be easily applied to the unbalanced case.

Initially, recall the unbalanced assignment problem (UAP) introduced in Section 3. Let $C \in \mathbb{R}^{n \times m}$ be a cost matrix with $n \geq m$, then UAP(C) is formulated as

$$\begin{aligned} \min_{X \in \mathbb{R}^{n \times m}} \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij} X_{ij} & & \\ \text{s.t.} \quad & \sum_{j=1}^{m} X_{ij} = 1 & & i = 1, \ldots, n \\ & \sum_{i=1}^{n} X_{ij} \leq 1 & & j = 1, \ldots, m \\ & X_{ij} \in \{0, 1\} & & i = 1, \ldots, n \quad j = 1, \ldots, m \end{aligned} \tag{3}$$

Before discussing the method some terminology is required. Let $\mathcal{X}_C$ denote the set of feasible integral solutions to UAP(C), that is all $X \in \mathbb{R}^{n \times m}$ satisfying constraints (3). Elements of the set $\mathcal{X}_C$ are also referred to as *assignments*. Proceeding with the fundamentals of the ranking method, we start with the concept of a node. Let $\{e_1, \ldots, e_r\}$ and $\{f_1, \ldots, f_l\}$ be sets of indices taken from $[n] \times [m]$, then the *node induced* by these indices is

$$\text{node}\{e_1, \ldots, e_r; f_1, \ldots, f_l\} = \{\, X \in \mathcal{X}_C \,:\, X_{e_1} = 1, \ldots, X_{e_r} = 1; X_{f_1} = 0, \ldots, X_{f_l} = 0\}$$

Essentially, a node is a set of assignments feasible for UAP(C) containing the edges $e_1, \ldots, e_r$ and not containing the edges $f_1, \ldots, f_l$. In general, a set of assignments is called a node if there exist sets of indices inducing it. For the sake of brevity, it is convenient to denote a node by the set of indices inducing it, that is $\{e_1, \ldots, e_r; f_1, \ldots, f_l\}$.

Given the node $N = \{e_1, \ldots, e_r; f_1, \ldots, f_l\}$, we are interested in the minimum cost assignment in N, which we denote X_N^* and its objective value by z_N. In the case that N is empty, the convenience $z_N = \infty$ is adopted. We may determine X_N^* by solving an instance of UAP with cost matrix obtained by selecting the rows and columns of C that are not fixed by the indices $e_1, \ldots, e_r$, and additionally setting the indices of forbidden edges $f_1, \ldots, f_l$ to infinity (or sufficiently large). The cost matrix obtained by this procedure is called the *remaining cost matrix at node* N. Let us illustrate this procedure by means of an example.

Consider the cost matrix

$$C = \begin{pmatrix} 2 & 4 & 5 & 2 \\ 3 & 3 & 1 & 6 \\ 7 & 8 & 0 & 9 \\ 9 & 2 & 6 & 2 \\ 1 & 5 & 4 & 1 \end{pmatrix}$$

with node $N = \{(3,3),(5,4);(\overline{4,1}),(\overline{1,2})\}$. In order to determine the minimum cost assignment in N we set $C_{41} = C_{12} = \infty$ and select the submatrix $\widetilde{C}$ of C consisting of rows $\{1,2,4\}$ and columns $\{1,2\}$, which is

$$\widetilde{C} = \begin{pmatrix} 2 & \infty \\ 3 & 3 \\ \infty & 2 \end{pmatrix}.$$

The optimal solution of UAP($\widetilde{C}$) is $\{(1,1),(3,2)\}$ which corresponds to the indices $\{(1,1),(4,2)\}$ in C. Therefore, the minimum cost assignment in N is $X_N^* = \{(1,1),(3,3),(4,2),(5,4)\}$ with $z_N = 5$.

We proceed with an essential node partitioning operation for the ranking algorithm. Let $N = \{e_1,\dots,e_r;f_1,\dots,f_l\}$ be a node and $X_N^* = \{e_1,\dots,e_r,g_1,\dots,g_{m-r}\}$ a corresponding minimum cost assignment. Naturally, the edges $g_1,\dots,g_{m-r}$ are all different from the restricted edges $f_1,\dots,f_l$. Consider the nodes

$$\begin{aligned} N_1 &= \{e_1,\dots,e_r;f_1,\dots,f_l,g_1\} \\ N_2 &= \{e_1,\dots,e_r,g_1;f_1,\dots,f_l,g_2\} \\ &\vdots \\ N_{m-r-1} &= \{e_1,\dots,e_r,g_1,\dots,g_{m-r-2};f_1,\dots,f_l,g_{m-r-1}\} \end{aligned}$$

Now let $X \in N$ be an assignment, then X either contains g_1 or it does not. If it does not then, $X \in N_1$. If it does, then X either contains g_2 as well, or it does not. If it does not, then $X \in N_2$, otherwise we proceed with the next edge. From this procedure it is clear that the sets $N_1,\dots,N_{m-r-1}$, partition the set $N \setminus \{X_N^*\}$. Note that X_N^* is not contained in any of these sets, which follows easily from the fact that every one of these sets restricts an edge from X_N^*. The procedure of constructing the sets $N_1,\dots,N_{m-r-1}$ is called *partitioning a node by its minimum cost assignment*.

4.6.1 Murty's algorithm

We are now ready for the solution method. Generally, the algorithm iteratively selects the node with the best optimal solution from a set of nodes and then partitions it according to its optimal solution. This iteratively selected node will be the next entry in the sequence $X_1,\dots,X_k$.

At first, there is the initialization procedure consisting of the construction of the first set of nodes. The

first assignment in the sequence may be determined by solving UAP using any available method. Let $X_1 = \{e_1, \ldots, e_m\}$ be this assignment. Now partition the set $\mathcal{X}_C \setminus \{X_1\}$ by its optimal assignment X_1, this gives the nodes

$$\begin{aligned} N_1 &= \{e_1\} \\ N_2 &= \{e_1; e_2\} \\ &\vdots \\ N_{m-1} &= \{e_1, \ldots, e_{m-2}; e_{m-1}\} \end{aligned}$$

Note that the set $\mathcal{X}_C \setminus \{X_1\}$ is not necessarily a node, hence partitioning it by its minimum cost assignment is a slight abuse of terminology. These nodes are collected in a set $L = \{N_1, \ldots, N_{m-1}\}$. Proceeding, the node with the best optimal objective is selected, that is

$$N = \operatorname*{argmin}_{M \in L} z_M.$$

Since the nodes in L partition the set $\mathcal{X}_C \setminus \{X_1\}$, it follows that $X_2 = X_N$. For the next iteration, the node $N = N_i$ is partitioned by its optimal assignment X_N into the sets $N_{i1}, \ldots, N_{id}$. The node N_i is then removed from L and the nodes $N_{i1}, \ldots, N_{id}$ are added. This means that in the next iteration the list of considered nodes is $(L \cup \{N_{i1}, \ldots, N_{id}\}) \setminus \{N\}$. Note that these nodes partition the set $\mathcal{X}_C \setminus \{X_1, X_2\}$, since $N_{i1}, \ldots, N_{id}$ is a partition of the set $N_i \setminus \{X_2\}$. This procedure continues until the k-best solution has been found. The entire algorithm is summarized in Algorithm 4.

Algorithm 4 Murty

$X_1 \leftarrow$ UAP(C)
$L \leftarrow$ PartitionNode($\mathcal{X}_C \setminus \{X_1\}$)
for $i = 1, \ldots, k$ **do**
  $N \leftarrow \operatorname{argmin}_{M \in L} z_M$
  $X_i \leftarrow X_N^*$
  $L_N \leftarrow$ PartitionNode($N \setminus \{X_N^*\}$)
  $L \leftarrow (L \cup L_N) \setminus \{N\}$
return $X_1, \ldots, X_k$

procedure PartitionNode(N)
  Let $N = \{e_1, \ldots, e_r; f_1, \ldots, f_l\}$ and $X_N^* = \{e_1, \ldots, e_r, g_1, \ldots, g_{m-r}\}$, then
    $N_1 = \{e_1, \ldots, e_r; f_1, \ldots, f_l, g_1\}$
    $N_2 = \{e_1, \ldots, e_r, g_1; f_1, \ldots, f_l, g_2\}$
    $\vdots$
    $N_{m-r-1} = \{e_1, \ldots, e_r, g_1, \ldots, g_{m-r-2}; f_1, \ldots, f_l, g_{m-r-1}\}$
  return $\{N_1, \ldots, N_{m-r-1}\}$

4.6.2 Complexity of Murty's algorithm

In conclusion, we analyze Murty's algorithm for complexity. Suppose that UAP(C) for $C \in \mathbb{R}^{n \times m}$ can be solved with complexity $O\big(f(n,m)\big)$. Murty's algorithm performs k-iterations in each of which a number of instances of UAP are solved. Assuming that the method keeps track of the minimum cost assignments of nodes in previous iterations, the number of instances of UAP to be solved is bound by the maximum cardinality of a partition of a node (by its minimal assignment). The maximum cardinality of such a partition is $m - 1$, in the case that there are no fixed edges in the corresponding node. Hence, each iteration can be resolved in $O\big(mf(n,m)\big)$, resulting in the overall complexity of $O\big(kmf(n,m)\big)$. A review of solution methods for UAP can be found in [3], which shows that state-of-the-art algorithms for UAP have a complexity of $O\big(n^3\big)$.

5 Identifying a track-truth link

Having introduced and modeled the problem of finding a track-truth link for a military simulation scenario, we can now turn to methods that can determine it. A fruitful perspective is to consider the task of identifying a track-truth link as an optimization problem, in which the objective is to identify a function $f : [0, t] \longrightarrow 2^E$ under the constraints that f is a track-truth link, introduced in 3.2. The idea here is that the optimal solution to the optimization problem coincides with the correct track-truth link. Inspired by the problem description, a suitable choice for an optimization problem representing the track-truth link problem is to represent it as an assignment problem (of some sort), introduced in Section 4.2. There are of course still variations between methods adopting an assignment problem approach, however they share a similar structure. Throughout this section, we consider a scenario with tracks $\mathcal{T} = \{T_1, \ldots, T_n\}$, observer partition $\{\mathcal{T}_1, \ldots, \mathcal{T}_l\}$, truths $\mathcal{D} = \{D_1, \ldots, D_m\}$, and alteration times $\{s_1, \ldots, s_k\}$.

5.1 Identification method

The method starts with three inputs: a *primary track* T, a *confusability range* R, and *predicted alteration times* $\tilde{s}_1, \ldots, \tilde{s}_k$. The primary track should be interpreted as the main track to be correlated. Since it is often not required to associate every track with its corresponding truth, only tracks that are confusable (possibly belong the same truth) with the primary track are selected. Simply put, we consider two tracks to be confusable if they lie within the confusability range R of each other. Since the exact alteration times are unknown to us, predicted alteration times are given as an argument to the method. The challenge of predicting alteration times accurately is discussed in Section 5.4.

Since the track-truth link changes at alteration times $s_1, \ldots, s_k$, it means that the track-truth link is invariant on each of the $k + 1$ intervals $[0, s_1], [s_1, s_2], \ldots, [s_k, t]$. Hence, the track-truth link consists of $k + 1$ sets of edges, corresponding to the correct track-truth link on each of these intervals. Each of our the methods presented will perform an optimization procedure on each of the intervals $[0, \tilde{s}_1], [\tilde{s}_1, \tilde{s}_2], \ldots, [\tilde{s}_k, t]$. The methods performed on each of the intervals will be identical.

5.1.1 Track and truth selection

Suppose that we are currently performing the optimization procedure on the interval $[\tilde{s}_p, \tilde{s}_{p+1}]$. At first, tracks are selected, whose indices (in the set $\mathcal{T}$) are collected in the set I_p. Initially, tracks that have been within the confusability range R of the primary track T at some point in time are selected, that is

$$i \in I_p \iff \exists\, \tau \in [\tilde{s}_p, \tilde{s}_{p+1}] \cap \chi_T : \|T_\rho(\tau) - T_{i,\rho}(\tau)\| \leq R.$$

Naturally, a track selected via this procedure might be confusable with a track that is not confusable with a different selected track. Therefore, the selection procedure is repeated for each of the selected tracks, proceeding until no new tracks are found. An example of this procedure is presented in Figure 3.

Proceeding, the distinct observers that have reported the selected tracks are determined, naming them $O_1, \ldots, O_h$. Note that it is not required that all of the l observers have reported a selected track during this interval, hence the necessity for h. An optimization procedure is executed separately for each of the h observers, which is done in consideration of property (iii) of Definition 3.2 (track-truth link). The explicit reason though will become more clear later. In the following, define I_{pq} to be the indices of tracks (in $\mathcal{T}$) reported by observer q in the interval p. Beforehand, truths are selected independently for each observer, and are collected in a set J_{pq}. A truth is selected if it has been within the confusability range of a track of the current observer, at some point in time. That is

$$j \in J_{pq} \iff \exists\, i \in I_{pq}\ \exists\, \tau \in [\tilde{s}_p, \tilde{s}_{p+1}] \cap \chi_{T_i} : \|T_{i,\rho}(\tau) - D_{j,\rho}(\tau)\| \leq R.$$

Now that the tracks and truths have been selected, we can proceed with the optimization procedure.

5.1.2 Computation of the track-truth link

Now that the tracks and truths have been selected, we can proceed with the optimization step. Essentially, the optimization procedure computes an optimal (according to some criterion) maximum cardinality matching in a subgraph of the track-truth graph. Recall that the track-truth graph (Section 3.2) is the bipartite graph $G(\mathcal{T}, \mathcal{D})$ with bipartite sets $V = \mathcal{T} \cup \mathcal{D}$ and edge set $E = \mathcal{T} \times \mathcal{D}$. Let H_{pq} be the subgraph of $G(\mathcal{T}, \mathcal{D})$ induced by vertices $\mathcal{T}_{pq} = \{T_i : i \in I_{pq}\}$ and $\mathcal{D}_{pq} = \{D_j : j \in J_{pq}\}$. We introduce binary variables $X_{ij} \in \{0, 1\}$ for $i \in I_{pq}$ and $j \in J_{pq}$ indicating whether track T_i originates from truth D_j. These variables can now be employed to formulate the optimization procedure.

Firstly, an issue needs to be dealt with regarding the correlation of truths. Property (ii) of a track-truth link asserts that every truth corresponds to at least one track. However, after the selection procedure, we are not certain that the selected truths all correspond to a track reported by the current observer $\mathcal{T}_q$. After all, if two entities are in proximity to one another, both might be reported with tracks of distinct observers.

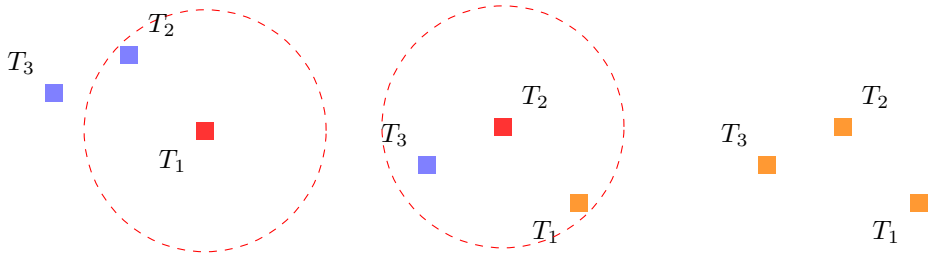


Figure 3: *Example of the track selection procedure for primary track T_1. Left: Current entity (red) is T_1, track T_2 (unselected are blue) lies within the confusability range and is selected for the next iteration. Center: Track T_3 lies in proximity of current track T_2 and is selected, track T_1 has already been selected (orange). Right: All tracks have been selected and the procedure is done.*

We make assumptions about this matter based on the number of selected tracks and truths. Firstly, in the case that the number of selected tracks $|\mathcal{T}_{pq}|$ exceeds the number of selected truths $|\mathcal{D}_{pq}|$ for the current observer, we assume that every truth corresponds to exactly one of the tracks. There are cases in which this assumption does not hold up, for instance when an abundance of ghost tracks has been selected. However, this is extremely unlikely. Secondly, in the case that $|\mathcal{T}_{pq}| \leq |\mathcal{D}_{pq}|$ we assume that each of the tracks corresponds to exactly one of the selected truths and additionally, that each truth correlates to at most one track. We formulate our assumptions as a result.

Assumption 5.1. *Let $\mathcal{T}_{pq}$ be the tracks and $\mathcal{D}_{pq}$ the truths obtained by the selection procedure for observer q in interval p. If $|\mathcal{T}_{pq}| \geq |\mathcal{D}_{pq}|$, then each truth corresponds to exactly one track. If $|\mathcal{T}_{pq}| \leq |\mathcal{D}_{pq}|$, then each track corresponds to exactly one truth and each truth to at most one track.*

It is now important to analyze how the properties of a track-truth link and our assumption, convert into constraints on the X_{ij}. The constraints rely on the number of selected tracks and truths. Initially, we derive constraints for the case that $|\mathcal{T}_{pq}| \geq |\mathcal{D}_{pq}|$. At first, property (i) of a track-truth link asserts that each track belongs to at most one truth. In terms of constraints, this means that

$$\sum_{j \in J_{pq}} X_{ij} \leq 1, \qquad \text{for all } i \in I_{pq}.$$

Then, property (iii) ascertains that distinct tracks reported by the same observer cannot correspond to the same truth. Additionally, property (ii) states that every truth object corresponds to at least one track. However, property (ii) does not guarantee that every selected truth object corresponds to a track of the current observer. Assumption 5.1 is required to derive this result. Together, these two properties imply that the degree of each truth in H_{pq} is exactly one, which is the constraint

$$\sum_{i \in I_{pq}} X_{ij} = 1, \qquad \text{for all } j \in J_{pq}.$$

Together, the constraints for the binary variables X_{ij} are

$$\begin{aligned} \sum_{j \in J_{pq}} X_{ij} &\leq 1, \quad i \in I_{pq} \\ \sum_{i \in I_{pq}} X_{ij} &= 1, \quad j \in J_{pq} \\ X_{ij} &\in \{0,1\}, \quad i \in I_{pq},\, j \in J_{pq} \end{aligned} \tag{10}$$

We can now continue with the case $|\mathcal{T}_{pq}| \leq |\mathcal{D}_{pq}|$. Property (i) in combination with Assumption 5.1 implies that the degree of each of the selected tracks is exactly one, which is enforced by the constraint

$$\sum_{j \in J_{pq}} X_{ij} = 1, \qquad \text{for all } i \in I_{pq}.$$

Assumption 5.1 also implies that the degree of each truth is at most one, yielding the constraint

$$\sum_{i \in I_{pq}} X_{ij} \leq 1, \qquad \text{for all } j \in J_{pq}.$$

The resulting constraints are therefore

$$\begin{aligned} \sum_{j \in J_{pq}} X_{ij} &= 1, \quad i \in I_{pq} \\ \sum_{i \in I_{pq}} X_{ij} &\leq 1, \quad j \in J_{pq} \\ X_{ij} &\in \{0,1\}, \quad i \in I_{pq},\, j \in J_{pq} \end{aligned} \tag{11}$$

At this stage, the constraints of our optimization procedure are defined for both cases. Unsurprisingly, these constraints are exactly the constraints of an assignment problem. All that is left is to specify what type of assignment problem is considered. For this, two options are proposed, which are an unbalanced assignment problem (UAP) and a lexicographic assignment problem (LexBAP). Recall the objectives of both these problems which are respectively to determine a maximum cardinality matching that: has a minimal sum of weights (UAP), and has a lexicographically minimal weight vector (LexBAP). Both of these approaches require a cost matrix C, which will be determined by the weight profiles discussed in Section 5.5.

After the optimization procedures have been performed for the distinct observers of the current interval, all that is left is to combine the results into a prediction for the track-truth link. In each interval, the set $E_{pq} = \{(T_i, D_j) \,:\, X_{ij} = 1\}$ represents the correlation of tracks reported by observer q in interval p, with truths. The predicted track-truth link E_p in interval p is now obtained by taking the union over all observers, that is $E_p = \bigcup_{q=1}^{h} E_{pq}$.

5.2 Motivation for UAP and LexBAP

For the optimization part of the track-truth link method, two options were proposed: UAP (3) and LexBAP (6). These criteria correspond respectively to a solution with a minimal sum of costs and a lexicographically minimal vector. Naturally, these two options require motivation through practical examples.

The intuition behind the UAP approach lies in minimizing the sum of errors. Given a cost matrix C representing errors between the tracks and truths, a logical choice for the track-truth link is the one that has the smallest sum of errors, i.e. the optimal solution of the unbalanced (or balanced) assignment problem. In this approach, there is no consideration of a maximum error in the track-truth link. As a consequence, an optimal track-truth link determined by UAP may contain a high error edge, which is then corrected (in terms of objective value) by low error edges. Since a high error edge is likely to be wrong, this approach might be less desirable.

Approaches that do not have this issue are the bottleneck assignment problem (BAP) and its lexicographical variant LexBAP. In a BAP (5) approach, the maximum error is minimized rather than the sum of errors. This resolves the issue of compensation in the UAP approach. However, instances of BAP usually have a large number of optimal solutions as there is no attention to solution behavior below the optimal value.

Therefore, a more appropriate method is LexBAP, which sorts the solution lexicographically according to its weight vector. Naturally, the optimal solution to LexBAP is also an optimal solution to BAP.

Unsurprisingly, it is common that for a given cost matrix C, both UAP(C) and LexBAP(C) yield the same optimal solution(s). It is interesting for our method to examine cases where the optimal solutions do not coincide since these are the situations where the choice between UAP and LexBAP matters in deciding between the right and wrong solutions. In Figure 4 a scenario is presented consisting of tracks T_1, T_2, T_3 and truths D_1, D_2, D_3, assuming no track-truth link as of yet. Assuming that the cost matrix C is constructed via distances, i.e. $C_{ij} = \|T_{i,\rho} - D_{j,\rho}\|$, the instances UAP($C$) and LexBAP($C$) do not share the same optimal solution.

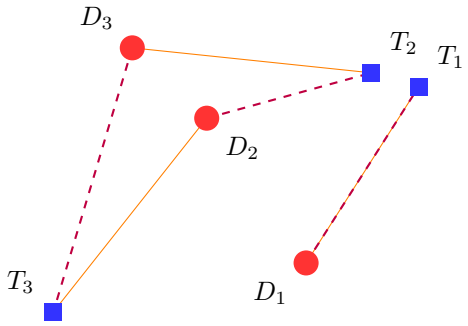


Figure 4: *Example instance with tracks T_1, T_2, T_3 and truths D_1, D_2, D_3 where UAP (purple, dotted) and LexBAP (orange, solid) give different solutions when using a cost matrix containing distances as weights.*

5.3 Analysis of method in edge cases

The method for the identification of a track-truth link defined above is theoretically sound. However, due to imperfections in the assumptions and rare occurrences in the data, there are cases in which the method does not perform as desired. Here we consider a few cases in which this happens and the associated solutions.

Two main occurrences in data can interfere with the solution method, which are respectively the selection of ghost tracks and undetected truths in the selection procedure. Recall that ghost tracks are tracks reported by the current observer, which are the result of simulation flaws or errors in detection. Therefore, ghost tracks do not correspond to any truth objects. Undetected truths, as the name suggests, are truths for which there does not exist a track of the present observer, that corresponds to it. There are two explanations for undetected truths. Firstly, there is the occurrence in military simulation that when multiple truths are close together far away from an observer, they are detected as a single track. Secondly, it is possible that a truth is not detected by any observer, but it is selected since it lies within the confusability

range of another track. This differs slightly from detecting multiple truths as one. An explanation for this is that the truth lies outside the range of the observer, but only barely as the observer has detected another track close to it. It could also be that the truth is obscured by an object (for example another object), meaning that the electromagnetic radiation cannot reach the entity, thus resulting in the non-detection of it.

The inclusion of ghost tracks and undetected truths may interfere with the track-truth link computation method because the method enforces the computation of a maximum cardinality matching for each observer. As a consequence, the track-truth link computed by our method may contain undesirable edges. An undesirable edge is an edge between a ghost track and a detected truth or between a non-ghost track and an undetected truth. To analyze how the selection of unwanted tracks and truths affects the solutions computed by our method, a case distinction between the inclusion of ghost tracks and/or undetected truths is required. In the case distinction, we assume that the confusability range of the method is sufficiently large, that is $R \geq \max\{R_1, \ldots, R_l\}$.

5.3.1 No ghost tracks, no undetected truths

Suppose for the first case that no ghost track or undetected truths are included in the selection process. Under the assumption that the confusability range of the method is at least as large as the confusability range of the present observer, we can conclude from property (iv) of a track-truth link (bounded track error) that the number of select tracks is equal to the number of selected truths. Naturally, this means that our method does not enforce undesirable edges.

5.3.2 At least one ghost track, no undetected truths

If there exists at least one selected ghost track, and there are no undetected truths, it is evident that the number of selected tracks is strictly larger than the number of selected truths. Again, this is under the assumption that the confusability range is chosen sufficiently large. In this case, the solution method computes a matching of cardinality equal to the number of selected truths. Naturally, it is desired that the proposed track-truth link does not include a ghost track in one of its edges. Annoyingly, this is of course a feasible outcome. Nevertheless, this outcome is unexpected and the method does not enforce the correlation of a ghost track with a detected truth.

5.3.3 No ghost tracks, at least one undetected truth

In the case of no ghost tracks and at least one undetected truth, the number of selected tracks is strictly smaller than the number of selected truths. Therefore, our method employs its alternate formulation to compute the track-truth link for the current observer. Similar to the last case, it is feasible that the method computes a track-truth link that contains an undetected truth in one of its edges. However, this is unexpected and the method does not enforce an edge containing an undetected truth.

5.3.4 At least one ghost track, at least one undetected truth

The last case to consider is that the selection procedure has included at least one ghost track and at least one undetected truth. In this case, it cannot be said that the number of tracks is strictly larger, smaller, or equal to the number of selected truths. For the sake of our analysis define the numbers t_{pq} of non-ghost tracks, g_{pq} of ghost tracks, d_{pq} of detected truths (for which there exists a track), and u_{pq} of undetected truths. Naturally, the cardinalities of tracks and truths correspond to the sums, that is $t_{pq} + g_{pq} = |I_{pq}|$ and $d_{pq} + u_{pq} = |J_{pq}|$.

Under the assumptions made above, it is guaranteed that an undesirable edge is included in the track-truth link computed by our method. Moreover, the number of such edges is equal to $\min\{g_{pq}, u_{pq}\}$. To see this, first assume that $|I_{pq}| \geq |J_{pq}|$. This means that the cardinality of the track-truth link is $t_{pq} + g_{pq}$, thus yielding g_{pq} undesirable edges. Contrarily, if $|I_{pq}| \leq |J_{pq}|$, the number of undesirable edges is u_{pq}. Since the confusability range is sufficiently large, property (iv) of a track-truth link guarantees that $t_{pq} = d_{pq}$. Thus, it follows that the number of undesirable edges is $\min\{g_{pq}, u_{pq}\}$.

5.3.5 More selected truths than tracks

As prescribed in the method, the assumptions and theory guarantee that the number of truths $|J_{pq}|$ in a scenario is smaller or equal to the number of tracks $|I_{pq}|$. There are however cases, where this assumption does not hold up. Firstly, there is the occurrence in military simulation that when multiple truths are close together far away from the observer, they are detected as a single track. When this happens, the number of selected truths likely exceeds the number of tracks. Secondly, it is possible that truths are not detected by any observer, but they are selected since they lie within the confusability range of another track. This differs slightly from detecting multiple truths as one. An explanation for this is that the truth lies outside the range of the observer, but only barely as the observer has detected another track close to it. It could also be that the truth is obscured by an object (for example another object), meaning that the electromagnetic radiation cannot reach the entity, thus resulting in the non-detection of it.

Whenever our method has selected more truths than tracks, we can be certain that one of these two issues has occurred. Our current assumptions of the track-truth link do not account for this situation. Hence, it is important to remedy the associated consequences the best we can. In the case that $|J_{pq}| \leq |I_{pq}|$, it is clear that the system of equations (10) no longer has solutions. Fortunately, this issue may be resolved by

altering constraints (10), into

$$\begin{aligned} \sum_{j \in J_{pq}} X_{ij} &= 1, \quad i \in I_{pq} \\ \sum_{i \in I_{pq}} X_{ij} &\leq 1, \quad j \in J_{pq} \\ X_{ij} &\in \{0,1\}, \quad i \in I_{pq},\, j \in J_{pq} \end{aligned} \tag{12}$$

For the rest, the method stays the same. One can view this change in constraints as mapping truths to tracks, rather than tracks to truths. The resulting set of edges now contains every track in $\mathcal{T}_{pq}$ exactly once, and every truth in $\mathcal{D}_{pq}$ at most once.

5.4 Alteration time estimation

One of the properties of a track-truth link is time-dependency, see Section 3.1 for a review. At certain moments during a scenario, the track-truth link may alter from one to another. We may recall from Section 3.1, that there were three main explanations for a change in the track-truth link: identity swapping, track correlation, and track decorrelation. These time points were previously introduced as alteration times. Since our solution method partitions the scenario time according to the alteration times, they must be estimated as accurately as possible. This section deals with this challenge.

At first, we consider estimating alteration times caused by track correlation. In its simplest form track correlation removes a track from the datalink if two tracks represent the same entity. The exact protocols determining the similarity of the tracks are confidential, though they are dependent on several properties such as position, velocity, and track quality (accuracy). When a datalink participant presumes two such tracks exist, he requests a correlation of the two via the J-series message J3.2 (correlation) over the datalink. If this request is accepted, one of the two tracks is dropped.

We elaborate on track correlation via an example. Consider a scenario containing a single fighter (red) labeled as D, and two frigates equipped with radar, labeled O_1 and O_2. Both frigates O_1 and O_2 are reporting fighter D under tracks T_1 and T_2 respectively. At a certain stage, the correlation protocol concludes that T_1 and T_2 both represent the same entity. As a consequence, track T_1 is kept and T_2 is dropped.

In the case of an alteration due to track correlation, we may approximate the alteration time using J-series messages. In Link-16, a J7.0 message (track management) is transmitted over the data link whenever a track is dropped. Since this message is transmitted at the same time (or shortly after), the receive time of this J-series message is a precise approximation of the alteration time. Therefore, they may determine the alteration times corresponding to track correlation.

Second, the estimation of alteration times due to track decorrelation is discussed. Track decorrelation can be perceived as the opposite of track correlation. To synchronize all the radar images of observers in a scenario, the observers compare local tracks with the tracks available on the datalink. Similar to the protocols in track correlation, there are protocols that determine whether local tracks correspond to tracks on the datalink. If a local track is correlated to a datalink track, nothing happens. On the other hand, if the observer has a local track for which there are no datalink tracks, the observer creates a new track and transmits it on the datalink.

There are two cases for the transmission of a new track. At first, there is the most probable explanation, which is that the entity (or entities) represented by the track has not been detected until this point. Alternatively, an entity may be no longer be tracked accurately by a different observer, due to range, simulation errors, or other reasons. At this point, the synchronization protocols of one observer may decide that a local track and datalink track (representing the same entity), are not the same. As a consequence, the observer creates a new track and transmits it on the datalink.

In both cases, the track-truth link is altered by the addition of a new track. Fortunately, the estimation of the corresponding alteration times is straightforward. Similar to track correlation, sending times of J-series messages can be employed to determine when the alteration has occurred. The J-series messages in question, are those containing positional information, which are the J3.x messages.

Thirdly, the estimation of alteration times due to identity swapping is treated. Recall that identity swapping is the alteration of the track-truth link obtained by swapping the correlated tracks of two truths. Identity swapping is caused primarily by radar noise, which causes difficulties in the distinction between the two entities. Out of the three causes of track-truth link alteration, identity swapping is the most challenging in determining the alteration time. What makes the swap times assessment so difficult, is that there are no reports (in the form of J-series or other) pertaining to the occurrence of an identity swap. Additionally, in analyzing scenario data, one can only conclude that a swap has occurred after some considerable time, making accurate estimation especially difficult.

A consequence of these factors is that accurate identity swap time estimation is nearly impossible. However, it is possible to determine a time interval in which the swap occurs. In practice, it is customary to determine the track-truth link at a time using visual inspection in a simulation data player, in which track and truth data are shown simultaneously over time. Annoyingly, this is not possible at all times due to discrepancies in track and truth positions. Therefore, the best we can do is determine by visual inspection the two times at which the respective track-truth links were believed to be correct, resulting in an interval in which the swap time is contained. Consequently, we can make a guess based on these interval boundaries, for example, the average of the two boundaries.

5.5 Weight Selection

Modeling track-truth linking as an assignment problem (of some sort) it is necessary to decide on what weight is chosen for an edge between track T and truth D. Most important for the motivation and selection of a weight profile is what joint properties of track T and truth D govern if T corresponds to D. This section describes and motivates a collection of reasonable weight profiles, which is a method for computing weights, for the track-truth linking problem.

Initially, a framework is suggested for the weight profiles. Let $\mathcal{T} = \{T_1, \ldots, T_n\}$ and $\mathcal{D} = \{D_1, \ldots, D_m\}$ be the set of tracks and truths respectively. Given that a track T_i corresponds to truth D_j, it is natural to model T_i as a noisy observation of D_j. In other words, given that $T_i \sim D_j$ there exists a random variable W_{ij} such that

$$T_{i,\rho} = D_{j,\rho} + W_{ij}.$$

At this stage, no assumptions are made about both the distribution of the W_{ij} and the possible dependence between them. The weight profiles discussed in this section make assumptions about the distribution of W_{ij} and independence and exploit it to accurately assess the likelihood that track and truth correspond to one another. The modeling assumption made above is easily justified by the errors made by radar systems and communications (which can be unpredictable).

5.5.1 Distance Weight Profile

An especially simple weight profile is the *Distance Weight Profile* (DW). The DW-profile assumes two things, the first of which is that all the W_{ij} are independent and identically distributed. The assumption of independence is rather strong as there is no reason why distinct instances of errors of radar detections and network errors influence one another. The identical distribution of the W_{ij} is a reasonable assumption for equal radar and network systems. Since distinct systems may have different noise profiles, this assumption will be weaker in scenarios with multiple types of radar. The second assumption is that

$$E[W_{ij}] = E[T_{i,\rho} - D_{j,\rho}] = 0, \quad T_i \sim D_j$$

which is based on the assumption that corresponding track and truth are expected to lie close together. Naturally, this assumption is strong as radar systems are designed to asses location as accurately as possible.

Using these assumptions a reasonable weight can be determined. According to the law of large numbers (the weak version suffices here), the sample mean of a sequence of independent identically distributed random variables converges to the expected value of the distribution (in probability). In the context of track T and truth D, given that T has time sequence $t_1, \ldots, t_k$, this is exactly

$$\frac{1}{k}\sum_{i=1}^{k} T_\rho(t_i) - D_\rho(t_i) \longrightarrow 0, \quad \text{in probability}$$

Therefore, a natural choice for weights is the norm of the sample mean of the differences between the track and truth, hence the weight function for the DW-profile is chosen as

$$C(T, D) = \left\| \frac{1}{k} \sum_{i=1}^{k} T_\rho(t_i) - D_\rho(t_i) \right\|.$$

Since the number k of observations varies from track to track, it is of importance here. Otherwise, it could be discarded.

5.5.2 Gaussian Weight Profiles

The second and third profiles for weights are the *Gaussian Weight Density* (GW-D) and *Gaussian Weight CDF* (GW-CDF) profiles, which assume a multivariate Gaussian distribution on the random variable W_{ij}. Additionally, this profile assumes that the noise variables W_{ij} are independent and identically distributed and that $E[W_{ij}] = 0$, as in the DW profile. The assumption that the noise follows a Gaussian distribution is a classical one that finds its roots in the central limit theorem.

We review the relevant properties of the multivariate Gaussian distribution. Let $X = (X_1, \ldots, X_l) \sim N(\boldsymbol{\mu}, \Sigma)$ be a random variable following the Gaussian distribution with mean $\boldsymbol{\mu}$ and positive semi-definite covariance matrix Σ. The covariance matrix Σ collects the covariances of the random variables X_i via $\Sigma_{ij} = \mathrm{Cov}(X_i, X_j)$. The probability density function of the multivariate Gaussian distribution is

$$\phi(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^l |\Sigma|}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right).$$

Here $|\Sigma|$ denotes the determinant of Σ. The cumulative distribution is obtained by integrating the density

$$\Phi(\boldsymbol{x}) = P(X \leq \boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^l |\Sigma|}} \int_{-\infty}^{x_1} \int_{-\infty}^{x_2} \cdots \int_{-\infty}^{x_k} \exp\left(-\frac{1}{2} (\boldsymbol{y} - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{y} - \boldsymbol{\mu}) \right) \mathrm{d}\boldsymbol{y}.$$

There are no closed-form expressions for the distribution function, though it can be estimated via numerical integration.

Given that the noise terms W_{ij} follow a Gaussian distribution, a weight can be determined that exploits this property. A reasonable choice for the weight between tracks T and truth D is the probability of observing $T_\rho - D_\rho$ under the assumption that $T_\rho - D_\rho \sim N(\mathbf{0}, \Sigma)$. If track T has been reported multiple times, using the mean of observations is beneficial. Computing this probability seems convenient, however, there are two difficulties at play here. Firstly, the covariance matrix is unknown to us, implying that computing exact probabilities according to the underlying Gaussian distribution is impossible. Secondly, since we are dealing with a continuous distribution, the probability of observing exactly $T_\rho - D_\rho$ is zero.

Let us resolve these issues. In the following suppose that track T has been observed at times $t_1, \ldots, t_k$ and that $T_i = T(t_i)$ and $D_i = D(t_i)$. Additionally, denote by $\overline{T_\rho} - \overline{D}_\rho$ the average of the differences

$T_{1,\rho} - D_{1,\rho}, \ldots, T_{k,\rho} - D_{k,\rho}$. To resolve the issue of the unknown covariance matrix, it seems attractive to employ estimation theory to obtain an estimate for Σ. However, to estimate Σ, the correct track-truth link is required. Therefore, estimation is not a feasible option. How unpleasant it may be, an educated guess of Σ might be the best result achievable. In practice, noise-information of the respective radar systems is known, allowing for an accurate guess of Σ. In the future, denote by Q the educated guess of Σ. The second issue can be resolved by either applying a density or distribution function approach. In the density approach, a Gaussian density function is evaluated at $\overline{T}_\rho - \overline{D}_\rho$, which gives an adequate prediction of the likelihood that the observations are samples from $N(0, \mathbf{\Sigma})$. Using the density function in this manner is common in estimation theory, where the density serves as the likelihood function [20]. Alternatively, the distribution function approach employs the probability of observing a more extreme result than $T_\rho - D_\rho$ as an indicator, which is a customary method in hypothesis testing [20].

In the density approach (GW-D), the probability density function of a Gaussian distribution is evaluated at the sample mean of the track and truth observations. The distribution of the sample mean of Gaussian random variables has reduced variance due to multiple observations. Specifically, the sample mean of k observations from distribution $N(\mathbf{0}, \Sigma)$ has distribution $N(\mathbf{0}, \Sigma/k)$. The density of the latter is evaluated at $\overline{T}_\rho - \overline{D}_\rho$, which is exactly

$$\phi\left(\overline{T}_\rho - \overline{D}_\rho\right) = \frac{1}{\sqrt{(2\pi)^l |\Sigma|}} \exp\left(-\frac{k}{2}\left(\overline{T}_\rho - \overline{D}_\rho\right)^\top \Sigma^{-1}\left(\overline{T}_\rho - \overline{D}_\rho\right)\right).$$

For the weights, the estimator Q required as Σ is unknown. Furthermore, the scalar multiplying the exponential is superfluous as all weights are multiplied by the same value. In a maximization approach, the weight for track T and truth D will then be

$$C(T, D) = \exp\left(-\frac{k}{2}\left(\overline{T}_\rho - \overline{D}_\rho\right)^\top Q^{-1}\left(\overline{T}_\rho - \overline{D}_\rho\right)\right).$$

In the distribution function approach (GW-CDF), the probability of observing a more extreme result is computed. As in the density approach, the sample mean is employed here to improve accuracy. Let us clarify what is meant by a more extreme result. Let X be an l-dimensional real random vector with expectation $E[X] = \mu$ and $x \in \mathbb{R}^l$. Then $x - \mu$ lies in one of the 2^l orthants, hence there exists a vector $\varepsilon \in \{-1, 1\}^l$ such that $\varepsilon_i (x_i - \mu_i) \leq 0$ for all $i = 1, \ldots, l$. The probability of observing a more extreme result than x for X is

$$P(X_1 \leq \varepsilon_1 x_1,\ X_2 \leq \varepsilon_2 x_2,\ \ldots,\ X_l \leq \varepsilon_l x_l)$$

In the one-dimensional case ($l = 1$), this is exactly $P(X \leq x)$ if $x \leq \mu$ and $P(X \geq x)$ if $x \geq \mu$. Denote by $p_X(x)$ the probability of observing for X a more extreme result than x. Under the assumption that X follows a Gaussian distribution, the Gaussian distribution function is employed to compute $p_X(x)$. The weight for track T and truth D in a maximization approach is then

$$C(T, D) = p_X(\overline{T}_\rho - \overline{D}_\rho), \quad X \sim N(0, \Sigma)$$

For a minimization approach, we can obtain weights by replacing each weight by its reciprocal.

6 Verification of solution method

Any theoretically derived method requires verification via testing, as is the case with the track-truth link method described in Section 5. This section establishes the framework for the verification and describes the verification method.

In theory, verification of the track-truth link method is quite simple: compare the track-truth link predicted by our method with the correct one for some military simulation scenarios. The scenarios used for verification should reflect those for which the method is designed, and should therefore be chosen (or designed) by experts who would apply the method in their line of work. An example of a valid verification scenario favored by experts is those exercised during the Joint Project Optic Windmill (JPOW) of the year 2023. Unfortunately, the underlying track-truth link is not monitored here and there are no methods capable of doing so accurately. As a consequence, our method can not be verified accurately through a comparison of the real and predicted track-truth links. The most reliable method for extracting the track-truth link from these scenarios is via visual inspection.

Alternatively, the method can be verified using the simulation environment JROADS (Joint Research on Air Defense Systems). Different from other military simulations, all systems and entities involved are simulated in the same environment. Consequently, the track-truth link can be monitored accurately for the duration of a scenario. Since JROADS is under development, it lacks some features and protocols that are required for tracking entities authentically (as it would in HIL-simulations or live situations). Since the inaccuracy of a track is predominantly determined by these features and protocols, the noise of a track and the corresponding truth generated by JROADS can not be as authentic as desired. JROADS does allow for simple noise profiles, which can be specified by the user. Additionally, JROADS controls the occurrence of identity swaps, by allowing the user to specify the quantity of swaps and when they occur (exact times or random).

Despite the tracking merits of JROADS, verification here is viable. In verification with JROADS, a constant diagonal Gaussian noise profile is assumed on the track noise (not to be mistaken with the Gaussian weight profile). Specifically, the position of a single track of a truth D is generated according to the three-dimensional Gaussian distribution $N(D, \sigma I)$, for some specified $\sigma \geq 0$.

6.1 Greedy method for benchmarking

To assess the quality of the methods presented, it is convenient to define a straightforward heuristic for the problem. The results produced by this heuristic, which we call the greedy method, will serve as a benchmark for the performance of our methods.

Recall that the solution methods for the track-truth link perform an optimization procedure for each ob-

server that has reported a track in the current interval. The optimization procedure consists of solving either an instance of UAP or LexBAP, both having the same set of feasible solutions. The greedy method computes a feasible solution by repeatedly adding the minimum cost edge in the graph, so that the collected set of edges in each iteration is a matching.

More explicitly, given a cost matrix $C \in \mathbb{R}^{n \times m}$ with $n \leq m$, the greedy method first selects the minimum entry C_{ij} in the cost matrix and fixes the associated binary X_{ij} variable to one. To prevent chosen entries in the same row or column in future iterations, the entire row and column (with the chosen entries' indices) are fixed to infinity. In the next iteration, the selection procedure is then repeated for the adjusted cost matrix. This procedure is repeated until n entries have been selected. The greedy method is summarized in Figure 5, in which $\mathbf{0}^{n \times m}$ denotes the zero-padded matrix of dimension $n \times m$. It is easily verified that the greedy method computes a maximum cardinality matching in linear time (updating an array is constant time).

Algorithm 5 Greedy matching

$X \leftarrow \mathbf{0}^{n \times m}$
for $k = 1, \ldots, n$ **do**
  $(i, j) \leftarrow \operatorname{argmin}\{C_{pq} : p \in [n],\, q \in [m]\}$
  $X_{ij} \leftarrow 1$
  $C \leftarrow$ UpdateCostMatrix(C, i, j)
return X

procedure UpdateCostMatrix(C, i, j)
  for $k = 1, \ldots, n$ **do**
    $C_{kj} \leftarrow \infty$
  for $k = 1, \ldots, m$ **do**
    $C_{ik} \leftarrow \infty$
  return C

6.2 Verification through JROADS

Verification via JROADs will be done via several scenarios each of which exercises difficulties that the method should account for. These difficulties include proximity formation flying, multi-track reporting by multiple observers, and identity swapping. The exact descriptions of the scenarios are given below. Each of the scenarios is also repeated for different noise levels of σ, which are also described in these descriptions.

6.2.1 Scenario 1: Formation Straight Single Observer

This scenario consists of five enemy fighters flying straight in a V-formation heading north during the entire scenario. There is a single frigate equipped with radar, reporting tracks of these entities. The duration of the scenario is six hundred seconds, throughout which no identity swaps occur. The scenario is executed $10\,000$ times for each of the noise levels $\sigma \in \{2500,\ 5000,\ 7500,\ 10\,000,\ 20\,000\}$. All three methods UAP, Greedy, and LexBAP are applied to this scenario with the distance weight (DW) profile. The duration of the scenario is ten minutes.

6.2.2 Scenario 2: Formation Straight Single Observers Two Swaps

This scenario is similar to Scenario 1, except for the occurrence of two identity swaps. The swap times are generated randomly from a uniform distribution, though accounting for a minimum separation time between consecutive swaps. The separation time between the randomly generated swap times is thirty seconds, meaning that there are at least thirty seconds between the two swaps. The noise levels are $\sigma \in \{2000, 2500, 3750, 5000\}$, each of which is simulated $10\,000$ times. The methods UAP and Greedy are applied to these scenarios, with the distance weight (DW) profile.

6.2.3 Scenario 3: Formation Straight Two Observers

This scenario is again similar to Scenario 1, though instead there are two frigates equipped with radar, rather than one. In this scenario the issue of multi-track reporting occurs, meaning that the two frigates will both report the entities. This method is repeated $10\,000$ times with the noise level $\sigma = 3750$. Again, the methods UAP and Greedy are applied to these scenarios with the distance weight (DW) profile.

6.3 Verification through JPOW

JPOW scenarios contain a large number of entities (both track and truth). Therefore, for the verification through JPOW, several situations from a scenario are selected which should reflect the difficulties of track and truth correlation. To achieve this, all situations have been assessed by an expert if they correctly represent these difficulties. The situations used for testing are briefly described in Table 3. Following confidentiality protocols regarding JPOW, these descriptions are not specific. To each of the situations, we apply the methods UAP, Greedy, and LexBAP, with a confusability range of $R = 3000$ and an arbitrary primary track out of the continuously existing tracks of the situation.

Situation	Intervals	Description
1	8	Formation of 4 fighters making a turn.
2	8	Formation of 6 fighters.
3	1	Pair of 2 helicopters in proximity flying straight.
4	1	Booster and warhead of ballistic missile.
5	3	Pair of 2 fighters in proximity.
6	1	Formation of 4 fighters flying straight.
7	2	Formation of 4 aircraft flying straight.
8	1	Formation of 3 helicopters.

Table 3: *Description of JPOW situations for verification.*

7 Results

This section presents the results of the verification described in Section 6. The results of the verification through JROADS are treated first, after which we continue with the verification through JPOW. The objective of the verification is to assess the performance of the presented methods in comparison with the Greedy method (as a benchmark).

7.1 Results of JROADS verification

The results of the methods in each of the three JROADS scenarios are first treated separately. Consequently, the results of the distinct scenarios are compared.

7.1.1 Scenario 1: Formation Straight Single Observer (FSSO)

The results of the methods applied to Scenario 1 FSSO are presented in Figures 5 through 8 (corresponding to the noise levels). Each of these figures shows histograms of the accuracies of each of the methods Greedy, LexBAP, and UAP, comparing them for each of the four noise levels. Additionally, the average accuracy over all the simulations is included. For an overview of the accuracies of the individual methods, Figures 9 through 11 show a histogram of accuracies of each of the noise level, including average accuracy as well.

For the noise level of $\sigma = 2500$, all of the three methods have almost identical performance, yielding the same average accuracy (rounded) of 0.99. When increasing the noise level to $\sigma = 5000$ we observe a decrease in average accuracies for all three methods. Here, UAP has the highest average accuracy of 0.89, followed by LexBAP (0.87) and Greedy (0.79). Moreover, the decrease for the Greedy method (of 0.20) is roughly twice as large as that of UAP (of 0.10) and LexBAP (of 0.12). Upon increasing the noise level even further to $\sigma = 7500$, the order of highest average accuracies remains the same. Similar to the previous noise level, a decrease of 0.20 (in average accuracy) is observed for the Greedy method. The decrease in UAP (0.18) and LexBAP (0.18) have increased concerning the prior noise level (0.10 and 0.12 respectively). In the final noise level with $\sigma = 10\,000$ average accuracy order remains the same. The average accuracies decrease further, yielding almost identical differences (concerning the previous level) of 0.12 (Greedy), 0.13 (LexBAP), and 0.13 (UAP).

From these results, we may derive that the difference between the average accuracies of the LexBAP and UAP methods is the same for the last three noise levels, with a value of 0.02. Furthermore, the percentage of simulations where these two methods give different results is at most 3.36% (for the last noise level), demonstrating that both methods have a similar distribution of accuracies. Therefore, it can be deduced that both LexBAP and UAP perform equivalently in Scenario 1 FSSO.

Most likely, the similarity of performance of the LexBAP and UAP methods is caused by the likeness in the objective of these methods. Recall that LexBAP and UAP methods aim to minimize the maximum cost edge and the sum of edge costs, respectively. Naturally, by minimizing the maximum of edge costs, the sum of edge costs is also reduced greatly. Specifically, the sum of edge costs in an optimal solution is at most the optimal value times the number of edges. On the other hand, minimizing the sum of edge costs frequently reduces the maximum cost of a selected edge, though no effective upper bound can be guaranteed on it.

Comparing UAP to the Greedy method, we observe identical average accuracies for the first noise level and differences 0.10, 0.12, and 0.11 for the last three noise levels, in favor of UAP. Additionally, UAP predicts the track-truth link with greater or equal accuracy than Greedy with percentages 99.9 %, 95.7%, 89.9%, and 85.6% in each of the four noise levels. Together, it is evident that UAP outperforms Greedy in Scenario 1 FSSO. Similar results are obtained for LexBAP, implying that LexBAP outperforms Greedy as well.

7.1.2 Scenario 2: Formation Straight Single Observer Two Swaps (FSSOTS)

Results for Scenario 2 FSSOTS are illustrated in Figures 13 through 16, with each figure corresponding to a different noise level. These figures include boxplots representing the average accuracies of the Greedy and UAP methods over the three intervals in each simulation run. Additionally, an overall average, called the interval average, is provided. Individual overviews for each method are shown in Figures 17 and 18.

At the first noise level ($\sigma = 2000$), both methods have similar interval averages, with UAP at 0.64 and Greedy at 0.61. The average accuracies of UAP are more centralized than those of Greedy, as indicated by their interquartile ranges (IQR) of 0.06 and 0.13, respectively. The third quartiles for both methods are identical at 0.66. At the second noise level ($\sigma = 2500$), the interval average difference between the methods increases slightly to 0.05 in favor of UAP. The IQR for UAP is 0.13, smaller than Greedy's 0.22, with the third quartile remaining equal.

At the third noise level ($\sigma = 3750$), UAP and Greedy have interval averages of 0.53 and 0.46, respectively. The IQRs are 0.14 for UAP and 0.20 for Greedy, with the third quartile for Greedy (0.53) being lower than that for UAP (0.60). At the highest noise level, the interval averages decrease further to 0.45 (UAP) and 0.38 (Greedy), with both methods having an IQR of 0.20.

The consistently smaller IQR for UAP compared to Greedy indicates that UAP's average accuracies are more centralized. Furthermore, the first and third quartiles for UAP are consistently higher than those for Greedy. These results imply that UAP generally achieves higher and more consistent average accuracies than Greedy. Across all noise levels, UAP's interval average surpasses that of Greedy, leading to the

conclusion that UAP outperforms Greedy in Scenario 2 FSSOTS.

7.1.3 Scenario 3: Formation Straight Two Observer (FSTO)

Figure 19 compares the UAP and Greedy methods in the form of a histogram of accuracies. In this scenario, UAP and Greedy perform with an average of 0.97 and 0.92, respectively. Out of all the simulation runs, UAP predicts the track-truth link with accuracy larger or equal to Greedy 97.7% percent of the time. Together, it is derived that UAP outperforms Greedy in Scenario 3 FSTO.

7.1.4 Weight Profile comparison

The final part of the JROADS results involves comparing the weight profiles discussed in Section 5.5. Three weight profiles were considered: Distance Weight (DW), Gaussian Weight Density (GW-D), and Gaussian Weight CDF (GW-CDF). To compare them, the UAP method was applied to Scenario 1 FSSO ($\sigma = 5000$) using cost matrices constructed from these three weight profiles. Figure 20 shows a histogram of the accuracies for each weight profile.

Among the three variants, the DW profile most frequently predicts the correct track-truth link, achieving an accuracy of 74.21% in the simulations. This is followed by the GW-D profile with 73.85% accuracy and the GW-CDF profile with 71.93% accuracy. The average accuracies differ by at most 0.01 among the three methods. Based on these results, the DW profile performs slightly better than both Gaussian weight profiles, though the difference is minimal. This outcome is unexpected, given that the discrepancy between track and truth positions follows an actual Gaussian distribution in these simulations.

The equivalent performance of the three methods is not surprising. For any track T and truth D, the three functions computing the corresponding cost matrix entry increase with the norm of the average differences $\left\|\overline{T_\rho} - \overline{D_\rho}\right\|$. Consequently, similar behavior among the methods is expected.

7.1.5 General JROADS results

By combining the results of the individual scenarios, we can make general statements regarding the effectiveness of the presented methods.

When comparing the accuracies of the methods in Scenario 1 (FSSO) and Scenario 2 (FSSOTS), it is evident that the performance in Scenario 2 is significantly worse. For a noise level of $\sigma = 2500$, both UAP and Greedy attain higher average accuracies in FSSO (both 0.99) than in FSSOTS, where they achieve accuracies of 0.61 and 0.56, respectively. Similarly, for a noise level of $\sigma = 5000$, UAP and Greedy in FSSO achieve accuracies of 0.89 and 0.79, respectively, compared to 0.45 and 0.38 in FSSOTS.

This performance gap between the two scenarios can be explained by the difference in the number of data points available. In FSSO, the methods can utilize all transmitted track and truth data. Conversely, in FSSOTS, the scenario is divided into three intervals, resulting in fewer data points for computing each of the three track-truth links. A larger sample size enhances the accuracy of the cost matrix, thus it is expected that the performance in FSSO would be superior to that in FSSOTS.

Comparing the performance of the methods in Scenarios 2 (FSSOTS) and 3 (FSTO) demonstrates that the methods handle the presence of multiple observers and swaps effectively, provided there are sufficient data points. The reduced performance in FSSOTS compared to FSSO was due to the reduction in data points. In FSTO, the methods UAP and Greedy exhibit similar performance to that observed in FSSO. Furthermore, there are no other reasons to believe that the method cannot handle track-truth link alteration and multiple observers, based on the JROADS simulations.

7.2 Results of JPOW verification

The results of the three methods for each of the JROADS situations are presented in Tables 4 through 8, where the single-interval situations are combined in a single table. Table 6 contains the accuracies of the single interval situations. Tables 4, 5, 7, and 8 contain the accuracies for the multi-interval situations, as well as the average accuracy.

In each of the situations, the three methods have almost identical accuracies. In situations 3 through 8, the accuracies are identical. In the first interval of situation 1, LexBAP performs with an accuracy of $\frac{2}{3}$, whereas UAP and Greedy both have an accuracy of zero. Since the accuracies are identical in the other intervals of situation 1, LexBAP yields a higher average accuracy of 0.91, than UAP (0.82) and Greedy (0.82). In all except one of the intervals of situation 2, the accuracies of the three methods are again identical. In interval 6, LexBAP has an accuracy of $\frac{5}{6}$, whereas UAP and Greedy have an accuracy of 1. Therefore, LexBAP has a smaller average accuracy here of 0.79, compared to UAP and Greedy (both 0.81).

In the JPOW situations, we observe the identical performance of UAP and Greedy, and a similar behavior of LexBAP. The accuracies of LexBAP vary from UAP and Greedy in only two intervals. Most likely, the instances corresponding to these intervals were similar to the one in Figure 4, though the exact reason was not analyzed. From the results, we conclude that UAP and Greedy have equal performance in the JPOW situations. There is not enough data to reliably assess the performance of LexBAP in comparison to Greedy and UAP.

8 Conclusion

The results presented in this thesis focus on evaluating three methods: Greedy, LexBAP, and UAP, against each other in various simulation scenarios using JROADS and JPOW data. The primary objective was to assess these methods' performance in predicting track-truth links, with the Greedy method serving as a benchmark.

In Scenario 1 FSSO, all methods initially achieve high accuracies, though with UAP consistently outperforming Greedy and LexBAP across increasing noise levels. LexBAP shows similar performance to UAP, though slightly worse. The similarity in performance between LexBAP and UAP can be explained by their similar objectives of minimizing maximum cost and sum of edge costs, respectively.

In Scenario 2 FSSOTS, UAP consistently achieves higher and more centralized average accuracies compared to Greedy across varying noise levels. The smaller interquartile range (IQR) for UAP indicates a more stable performance, which suggests that UAP performs better than Greedy in handling track-truth link alterations. Additionally, it shows that UAP copes better with a smaller number of data points.

In FSTO, UAP again outperforms Greedy with higher average accuracies, demonstrating its effectiveness in scenarios involving multiple observers. Comparing different weight profiles (DW, GW-D, GW-CDF) using the UAP method in FSSO ($\sigma = 5000$), the DW profile shows marginally better accuracy than Gaussian weight profiles, despite the Gaussian distribution on the track-truth error.

Across scenarios, the performance of methods varies. FSSO consistently yields higher accuracies compared to FSSOTS, which is likely caused by the availability of more data points. The methods generally handle multiple observers and track-truth link alterations effectively when the number of data points is sufficient. In JPOW situations, UAP and Greedy exhibit identical accuracies across the situations. LexBAP shows similar performance to UAP and Greedy in most intervals, with minor variations in a few cases.

Based on the results obtained from JROADS and JPOW verifications, several conclusions can be drawn regarding the performance of the Greedy, LexBAP, and UAP methods. Firstly, UAP consistently outperforms Greedy, regardless of the scenario or situation. UAP outperforms LexBAP as well (except for one JPOW interval), though only by a slight margin.

Secondly, the performance of UAP and Greedy is greatly affected by the number of data points available, though UAP is less sensitive to it than Greedy. Most likely, the same can be said for LexBAP, though due to the time-demanding factor of the simulations, data for LexBAP was not obtained. In future work, it will be valuable to investigate weight profiles for the current framework, that perform with high accuracy even with a limited amount of track-truth data. Such weight profiles would be incredibly valuable in track-truth

correlation, as the performance of the current methods would increase in scenarios with a quickly altering track-truth link.

One potential approach for developing such a weight profile is to incorporate an entity's velocity and acceleration. Initially, this might seem redundant, as entities in close proximity for the duration of a scenario will likely have similar velocities and accelerations, as otherwise, their positions would diverge. However, this relationship becomes less true over shorter intervals.

9 Appendix

9.1 Figures Scenario 1 FSSO

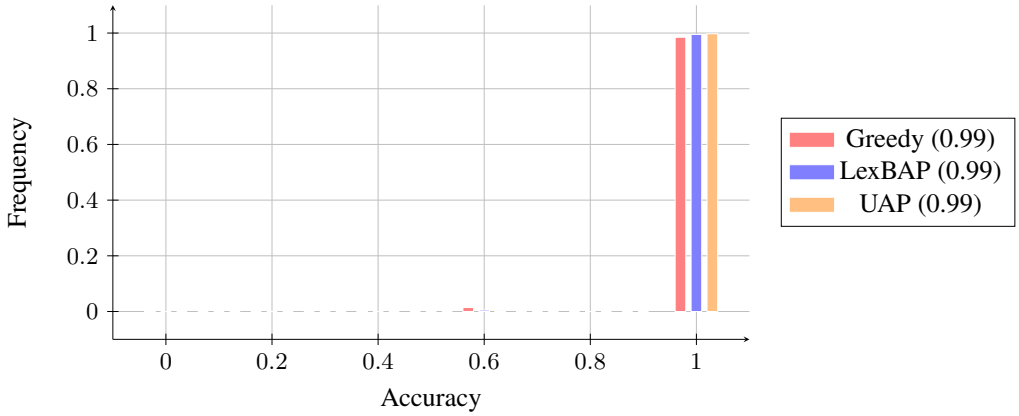


Figure 5: *Histogram of accuracies and averages comparing Greedy, LexBAP, and UAP in Scenario 1 FSSO ($\sigma = 2500$, 10 000 runs)*

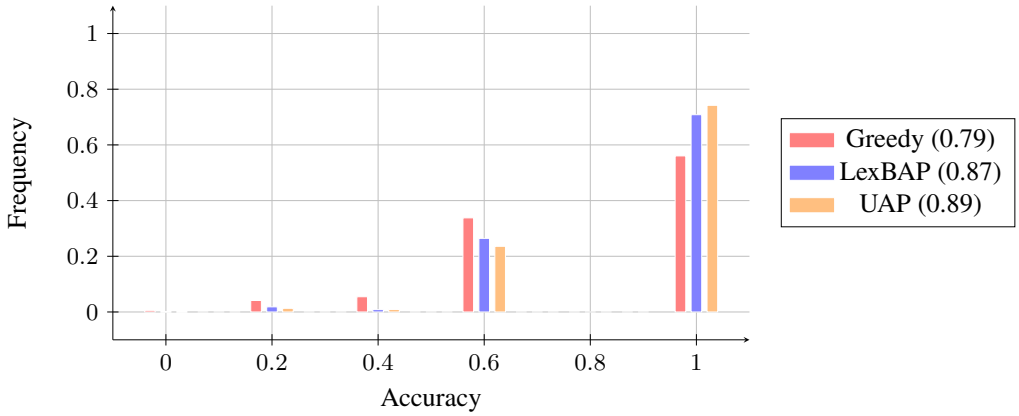


Figure 6: *Histogram of accuracies and averages comparing Greedy, LexBAP, and UAP in Scenario 1 FSSO ($\sigma = 5000$, 10 000 runs)*

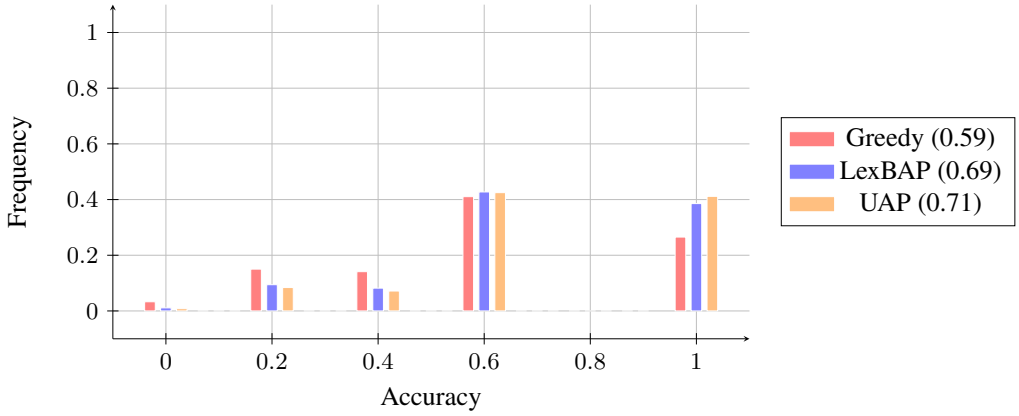


Figure 7: *Histogram of accuracies and averages comparing Greedy, LexBAP, and UAP in Scenario 1 FSSO ($\sigma = 7500$, 10 000 runs)*

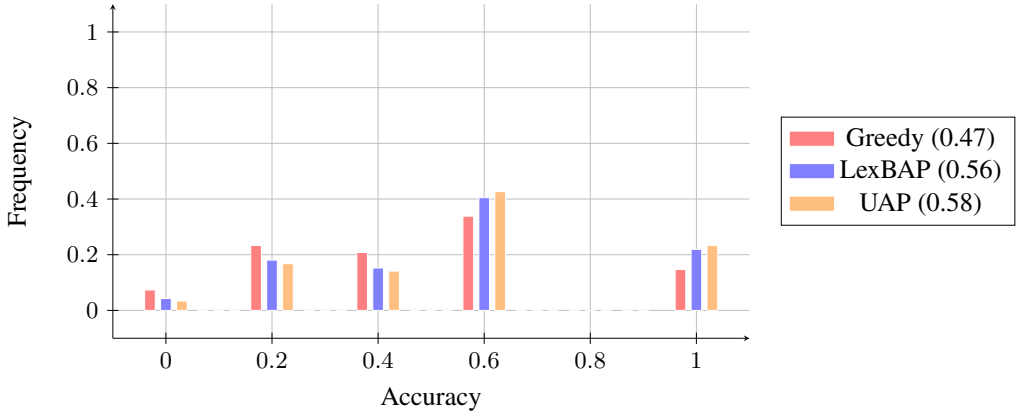


Figure 8: *Histogram of accuracies and averages comparing Greedy, LexBAP, and UAP in Scenario 1 FSSO ($\sigma = 10\,000$, 10 000 runs)*

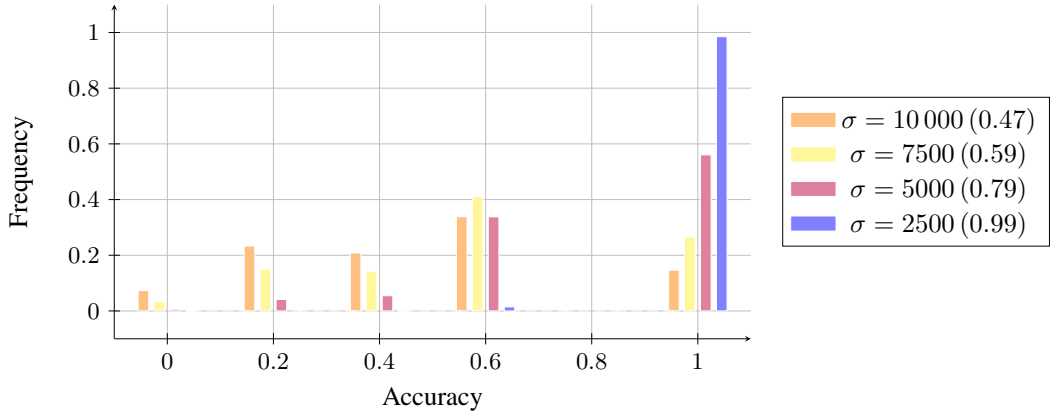


Figure 9: *Histogram of accuracies for Greedy applied to Scenario 1 FSSO for distinct noise levels (*10 000 *runs)*

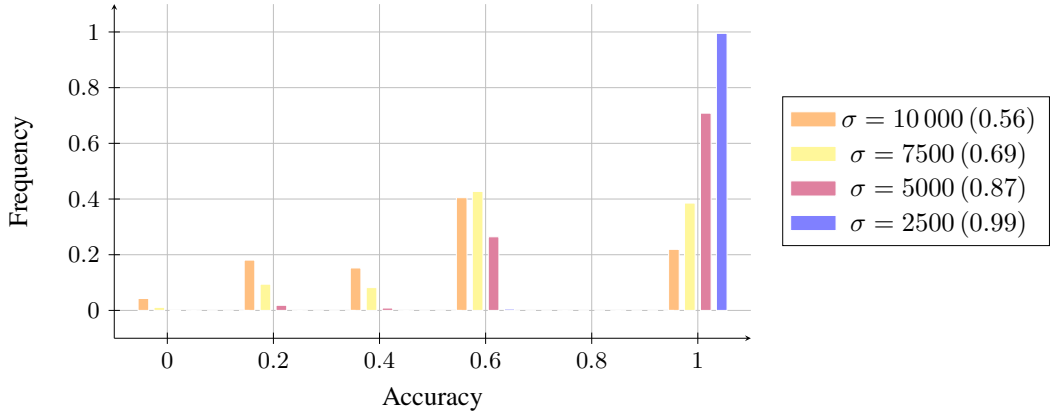


Figure 10: *Histogram of accuracies for LexBAP applied to Scenario 1 FSSO for distinct noise levels (*10 000 *runs)*

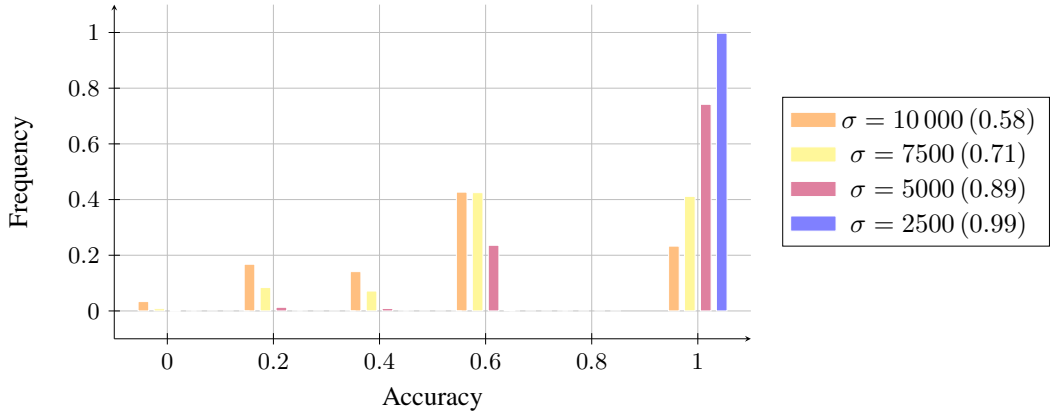


Figure 11: *Histogram of accuracies for UAP applied to Scenario 1 FSSO for distinct noise levels (*10 000 *runs)*

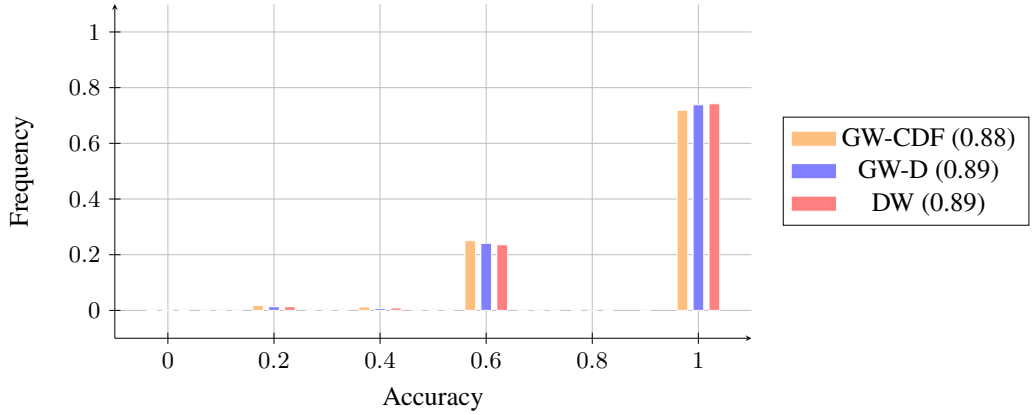


Figure 12: *Histogram of accuracies and averages comparing the Distance (DW), Gaussian Density (GW-D), and Gaussian CDF (GW-CDF) weight profiles in Scenario 1 FSSO ($\sigma = 5000$, 10 000 runs) with UAP method*

9.2 Figures Scenario 2 FSSOTS

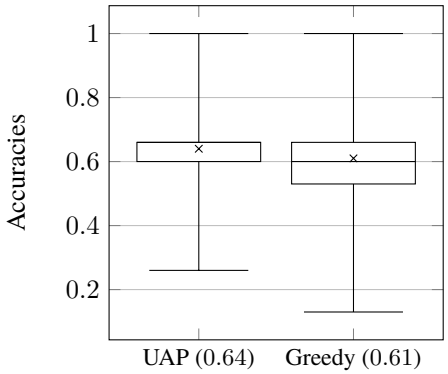


Figure 13: *Accuracy comparison of UAP and Greedy methods applied to Scenario 2 FSSOTS (σ = 2000, 10000 runs)*

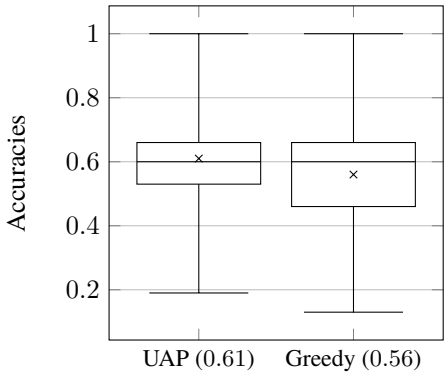


Figure 14: *Accuracy comparison of UAP and Greedy methods applied to Scenario 2 FSSOTS (σ = 2500, 10000 runs)*

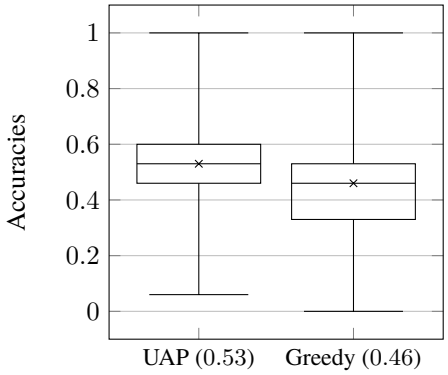


Figure 15: *Accuracy comparison of UAP and Greedy methods applied to Scenario 2 FSSOTS ($\sigma = 3750$, 10000 runs)*

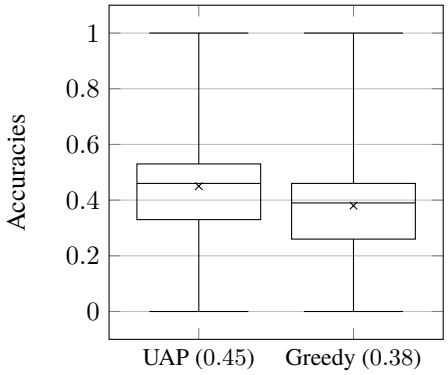


Figure 16: *Accuracy comparison of UAP and Greedy methods applied to Scenario 2 FSSOTS ($\sigma = 5000$, 10000 runs)*

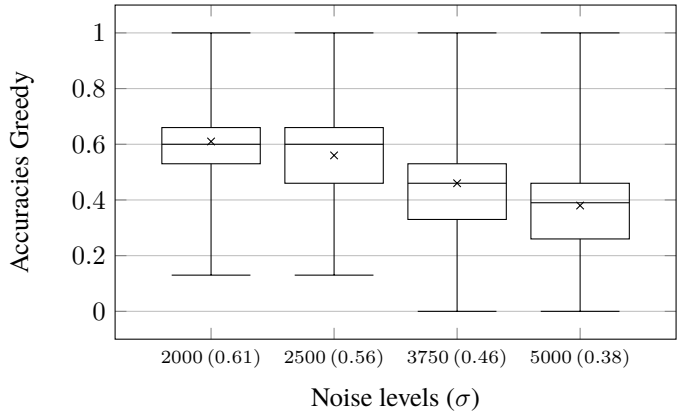


Figure 17: *Boxplot of accuracies of Greedy method applied to Scenario 2 FSSOTS with different noise levels.*

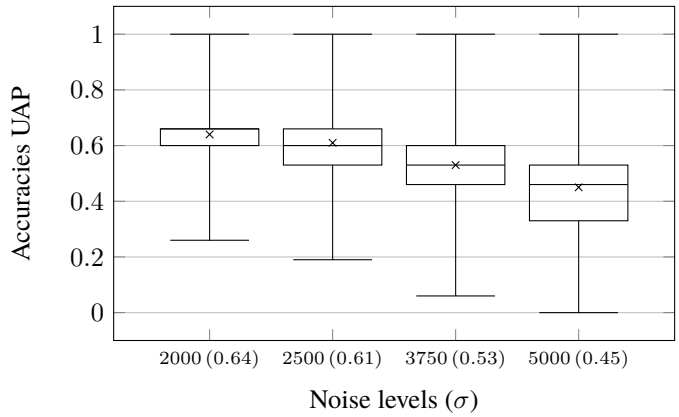


Figure 18: *Boxplot of accuracies of UAP method applied to Scenario 2 FSSOTS with different noise levels (for $\sigma = 2000$ the median coincides with the third quartile).*

9.3 Figures Scenario 3 FSTO

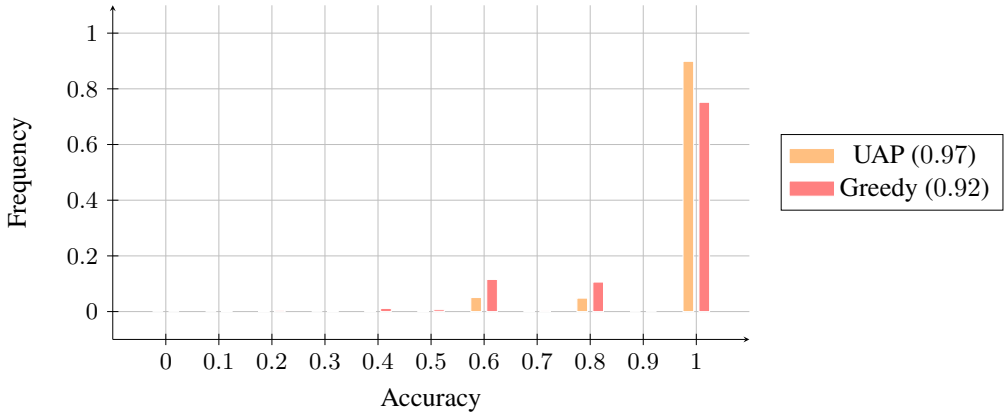


Figure 19: *Histogram of accuracies for UAP and Greedy applied to Scenario 3 FSTO* ($\sigma = 3750$, $10\,000$ *runs)*

9.4 Tables JPOW Situations

Interval	UAP	Greedy	LexBAP
1	0	0	$\frac{2}{3}$
2	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{3}{4}$
3	1	1	1
4	1	1	1
5	1	1	1
6	$\frac{5}{6}$	$\frac{5}{6}$	$\frac{5}{6}$
7	1	1	1
8	1	1	1
Average	0.82	0.82	0.91

Table 4: *Accuracies of methods UAP, Greedy, and LexBAP on JPOW situation 1.*

Interval	UAP	Greedy	LexBAP
1	0	0	0
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	1	$\frac{5}{6}$
7	$\frac{4}{5}$	$\frac{4}{5}$	$\frac{4}{5}$
8	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{2}{3}$
Average	0.81	0.81	0.79

Table 5: *Accuracies of methods UAP, Greedy, and LexBAP on JPOW situation 2.*

Situation	UAP	Greedy	LexBAP
3	1	1	1
4	1	1	1
6	1	1	1
8	1	1	1

Table 6: *Accuracies of methods UAP, Greedy, and LexBAP on single-interval JPOW situations (3,4,6,8).*

Interval	UAP	Greedy	LexBAP
1	1	1	1
2	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
3	1	1	1
Average	0.83	0.83	0.83

Table 7: *Accuracies of methods UAP, Greedy, and LexBAP on JPOW situation 5.*

Interval	UAP	Greedy	LexBAP
1	$\frac{4}{5}$	$\frac{4}{5}$	$\frac{4}{5}$
2	1	1	1
Average	0.9	0.9	0.9

Table 8: *Accuracies of methods UAP, Greedy, and LexBAP on JPOW situation 7.*

References

[1] H. Alt et al. "Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m/\log n})$". In: *Inform. Process. Lett.* 37.4 (1991), pp. 237–240. ISSN: 0020-0190,1872-6119. DOI: 10.1016/0020-0190(91)90195-N. URL: https://doi.org/10.1016/0020-0190(91)90195-N.

[2] Claude Berge. "Two theorems in graph theory". In: *Proc. Nat. Acad. Sci. U.S.A.* 43 (1957), pp. 842–844. ISSN: 0027-8424. DOI: 10.1073/pnas.43.9.842. URL: https://doi.org/10.1073/pnas.43.9.842.

[3] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2009, pp. xx+382. ISBN: 978-0-898716-63-4. DOI: 10.1137/1.9780898717754. URL: https://doi.org/10.1137/1.9780898717754.

[4] Joseph Cheriyan, Torben Hagerup, and Kurt Mehlhorn. "Can a maximum flow be computed in $o(nm)$ time?" In: *Automata, languages and programming (Coventry, 1990)*. Vol. 443. Lecture Notes in Comput. Sci. Springer, New York, 1990, pp. 235–248. ISBN: 0-387-52826-1. DOI: 10.1007/BFb0032035. URL: https://doi.org/10.1007/BFb0032035.

[5] Defensie. "JPOW: Europa's grootste lucht- en raketverdedigingsoefening". In: (2023). URL: https://magazines.defensie.nl/defensiekrant/2023/11/02_jpow_11.

[6] E. A. Dinic. "An algorithm for the solution of the problem of maximal flow in a network with power estimation". In: *Dokl. Akad. Nauk SSSR* 194 (1970), pp. 754–757. ISSN: 0002-3264.

[7] Northrop Grumman. *Understanding voice and data link networking*. Ed. by Daniel Akers. 2014.

[8] A. J. Hoffman and J. B. Kruskal. "Integral boundary points of convex polyhedra". In: *Linear inequalities and related systems*. Vol. no. 38. Ann. of Math. Stud. Princeton Univ. Press, Princeton, NJ, 1956, pp. 223–246.

[9] John E. Hopcroft and Richard M. Karp. "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs". In: *SIAM J. Comput.* 2 (1973), pp. 225–231. ISSN: 0097-5397. DOI: 10.1137/0202019. URL: https://doi.org/10.1137/0202019.

[10] IEEE. "IEEE Standard for Distributed Interactive Simulation–Application Protocols". In: *IEEE Std 1278.1-2012 (Revision of IEEE Std 1278.1-1995)* (2012), pp. 1–747. DOI: 10.1109/IEEESTD.2012.6387564.

[11] Alexander V Karzanov. "An exact estimate of an algorithm for finding a maximum flow, applied to the problem on representatives". In: *Problems in Cybernetics* 5 (1973), pp. 66–70.

[12] Henry Martyn Mulder. "Julius Petersen's theory of regular graphs". In: vol. 100. 1-3. Special volume to mark the centennial of Julius Petersen's "Die Theorie der regulären Graphs", Part I. 1992, pp. 157–175. DOI: 10.1016/0012-365X(92)90639-W. URL: https://doi.org/10.1016/0012-365X(92)90639-W.

[13] Katta G. Murty. “Letter to the Editor—An Algorithm for Ranking all the Assignments in Order of Increasing Cost”. In: *Operations Research* 16.3 (1968), pp. 682–687. DOI: 10.1287/opre.16.3.682. eprint: https://doi.org/10.1287/opre.16.3.682. URL: https://doi.org/10.1287/opre.16.3.682.

[14] NATO. “Interoperability: connecting forces”. In: (2023). URL: https://www.nato.int/cps/en/natohq/topics_84112.htm.

[15] NATO. “NATO Integrated Air and Missile Defence”. In: (2023). URL: https://www.nato.int/cps/ie/natohq/topics_8206.htm.

[16] Allied Air Command Public Affairs Office. “LARGEST EUROPEAN INTEGRATED AIR AND MISSILE DEFENCE EXERCISE ENDS”. In: (2023). URL: https://ac.nato.int/archive/2023/JPOW23_ends.

[17] Julius Petersen. “Die Theorie der regulären graphs”. In: *Acta Math.* 15.1 (1891), pp. 193–220. ISSN: 0001-5962,1871-2509. DOI: 10.1007/BF02392606. URL: https://doi.org/10.1007/BF02392606.

[18] Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience Series in Discrete Mathematics. A Wiley-Interscience Publication. John Wiley & Sons, Ltd., Chichester, 1986, pp. xii+471. ISBN: 0-471-90854-1.

[19] P. T. Sokkalingam and Y. P. Aneja. “Lexicographic bottleneck combinatorial problems”. In: *Oper. Res. Lett.* 23.1-2 (1998), pp. 27–33. ISSN: 0167-6377,1872-7468. DOI: 10.1016/S0167-6377(98)00028-5. URL: https://doi.org/10.1016/S0167-6377(98)00028-5.

[20] A. van der Vaart, M. Jonker, and F. Bijma. *An Introduction to Mathematical Statistics*. Amsterdam University Press, 2017. ISBN: 9789048536115. URL: https://books.google.nl/books?id=KMokDwAAQBAJ.

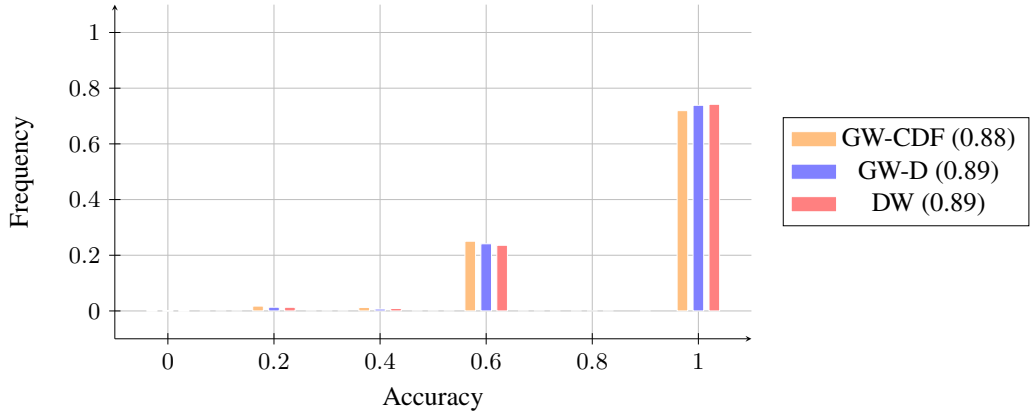


Figure 20: *Histogram of accuracies and averages comparing the Distance (DW), Gaussian Density (GW-D), and Gaussian CDF (GW-CDF) weight profiles in Scenario 1 FSSO ($\sigma = 5000$, $10\,000$ runs) with UAP method*