Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

Master's Thesis

OPTIMIZATION OF TUNING PLANS FOR A PASSIVE SURVEILLANCE SYSTEM

by

Bc. JAN PIKMAN

Study programme: Open Informatics
Specialization: Artificial Intelligence

Prague, May 2024

Supervisor:
doc. Ing. PŘEMYSL ŠŮCHA, Ph.D.
Department of Control Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo náměstí 13
120 00, Prague 2
Czech Republic

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Pikman Jan**	Personal ID number:	**492098**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Computer Science**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence**		

II. Master’s thesis details

Master’s thesis title in English:

Optimization of Tuning Plans for a Passive Surveillance System

Master’s thesis title in Czech:

Optimalizace tvorby ladicího plánu pro pasivní sledovací systém

Guidelines:

This thesis deals with the design of an algorithm for automatic tuning of a passive surveillance system. The algorithm has to find a tuning plan for specific frequency bands, so that it can search for new targets and then track their position. The goal is to achieve an efficient use of the receivers so that the system can track as many targets as possible while detecting new targets quickly enough.
1) Conduct a literature review in areas of scheduling for radars and the set cover problem.
2) Learn the principle of receivers tuning and target tracking in passive surveillance systems.
3) Analyze the computational complexity of the tuning plan generation problem and try to find connections with existing optimization problems.
4) Propose an algorithm for determining for each target how often should be tuned, designing receiver tuning setups, and creating a tuning plan.
5) Implement the proposed algorithm in Python, test it using simulation and evaluate the quality of the proposed solution.

Bibliography / sources:

Skolnik, M. An introduction and overview of radar. Radar Handbook, volume 3, 2008: pp. 1–1.
Kulmon, P. Bayesian Deghosting Algorithm for Multiple Target Tracking. In 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), 2020, pp. 367–372, doi:10.1109/MFI49285.2020.9235215.
Kulmon, P. Suja, J. and Benko, M. Scheduling of Multi-Function Sensor, in IEEE Transactions on Radar Systems, vol. 1, pp. 729-739, 2023, doi: 10.1109/TRS.2023.3335208.
Sun, J.; Yi, W.; Varshney, P. K.; et al. Resource Scheduling for Multi-Target Tracking in Multi-Radar Systems With Imperfect Detection. IEEE Transactions on Signal Processing, volume 70, 2022: pp. 3878–3893, doi:10.1109/TSP.2022.3191800.
Briheche, Y.; Barbaresco, F.; Bennis, F.; et al. Theoretical complexity of grid cover problems used in radar applications. Journal of Optimization Theory and Applications, volume 179, no. 3, 2018: pp. 1086–1106.
Caprara, A.; Toth, P.; Fischetti, M. Algorithms for the set covering problem. Annals of Operations Research, volume 98, no. 1-4, 2000: pp. 353–371.
Butman, A.; Hermelin, D.; Lewenstein, M.; et al. Optimization problems in multiple-interval graphs. ACM Transactions on Algorithms (TALG), volume 6, no. 2, 2010: pp. 1–18.

Name and workplace of master’s thesis supervisor:

doc. Ing. Přemysl Šůcha, Ph.D. Department of Control Engineering FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **25.01.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

doc. Ing. Přemysl Šůcha, Ph.D. Supervisor’s signature	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

First of all, I would like to express my utmost gratitude to my supervisor Associate Professor Přemysl Šůcha for his advice, feedback, knowledge, and enthusiasm without which this thesis would not be the same. My thanks also go to Professor Zdeněk Hanzálek for introducing me to the field of combinatorial optimization and for an opportunity to work in cooperation with ERA a.s. on a fascinating topic of optimization of passive radars.

Hlavně bych chtěl poděkovat svým rodičům za jejich neutuchající lásku a podporu. Rád bych také vyjádřil vděčnost své sestře, spolubydlícím a všem ostatním, kteří mi zlepšovali náladu a podporovali mě během studia.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, ...

Bc. Jan Pikman

Abstract

Passive radar systems are used for covert detection and tracking of transmitting objects over hundreds of kilometers. Unfortunately, these systems have a limited amount of receivers and therefore can simultaneously monitor only a fraction of frequencies of interest. This master's thesis proposes an algorithm for fast construction of optimized schedules, called tuning plans, that determine which frequencies will be observed by each receiver at a given time. An essential part of the algorithm is the optimization of receiver configurations using a newly formulated multiple-interval containment problem (MICntP) which is similar to the set cover problem over interval-like objects. We also study the theoretical complexity of different MICntPs based on imposed constraints. Finally, we analyze the dependence of the algorithm's run-time on properties and the number of randomly generated input requests during a simplified scenario.

Keywords:

optimization, passive radar, VERA-NG, scheduling, optimization of observed frequencies, multiple-intervals, set cover problem, multiple-interval containment problem.

Abstrakt

Pasivní radarové systémy se používají k detekci a sledování vysílajících objektů na vzdálenost stovek kilometrů. Tyto systémy mají bohužel omezený počet přijímačů, a proto mohou současně sledovat pouze zlomek požadovaných frekvencí. Tato magisterská práce se zabývá návrhem algoritmu pro rychlou konstrukci optimalizovaných rozvrhů, tzv. ladicích plánů, určujících, které frekvence budou v daném čase sledovány jednotlivými přijímači. Podstatnou částí algoritmu je optimalizace konfigurací přijímačů pomocí nově formulovaného problému obsažení vícenásobných intervalů (MICntP), který se podobá set cover problému s objekty připomínající intervaly. Také zkoumáme teoretickou složitost různých MICntP na základě zavedených omezení. Nakonec analyzujeme závislost doby běhu algoritmu na vlastnostech a počtu náhodně generovaných vstupních požadavků.

Klíčová slova:

optimalizace, pasivní radar, VERA-NG, rozvrhování, optimalizace monitorovaných frekvencí, vícenásobné intervaly, set cover problém, problém obsažení vícenásobných intervalů.

Contents

List of Figures

List of Algorithms

List of Abbreviations and Symbols

Number Sets

$\mathbb{N}$	Natural numbers set
$\mathbb{R}$	Real numbers set
$\mathbb{R}_{\geq 0}$	Non-negative real numbers set

Radar Terminology

ELINT	Electronic intelligence
ESM	Electronic support measure
LPI	Low probability of intercept
MRS	Multi-radar system
MTT	Multi-target tracking
PARN	Phased array radar network
PCL	Passive coherent location
PCR	Passive covert radar
PESA	Passive phased array
TDOA	Time difference of arrival

Problems

DHSP	Demand hitting set problem
GSCP	Geometric set cover problem
HSP	Hitting set problem

MICntP	Multiple-Interval containment problem
MC	Multicover problem
R3-SAT	Boolean satisfiability problem in which each variable appears exactly three times
SAT	Boolean satisfiability problem
SCP	Set cover problem
WDHSP	Weighted demand hitting set problem
WSCP	Weighted set cover problem

Algorithms

B&B	Branch and Bound
ED	Earliest deadline
EST	Earliest start time
GA	Genetic algorithm
ILP	Integer linear programming
JRAPS	Joint radar assignment and power scheduling
LP	Linear programming
NSGA-II	Non-dominated sorting genetic algorithm II
RSST-EST	Random shifted start time - earliest start time
ZMFD	Zoutendijk method of feasible directions

Intervals and Interval Sets

$l(i)$	Left endpoint of single interval i
$r(i)$	Right endpoint of single interval i
$\|i\|$	Size of single interval i, equal to $r(i) - l(i)$
$\mathbb{I}$	Set of unions of any number of single intervals
1-$\mathbb{I}$	Set of single intervals
t-$\mathbb{I}$	Set of unions of up to t single intervals

Passive Radar System

$\mathbf{SN}$	Collection of the system's sensor nodes
SN_i	i-th system's sensor node, $\mathrm{SN}_i \in \mathbf{SN}$
$\mathbf{R}_i$	Collection of receivers belonging to the i-th sensor node
$R_{i,j}$	j-th receiver belonging to the i-th sensor node, $\mathrm{R}_{i,j} \in \mathbf{R}_\mathrm{i}$
$C_{i,j}$	Configuration of receiver $R_{i,j}$

$f(C_{i,j})$	Multiple-Interval that describes which frequencies are monitored by the configuration $C_{i,j}$
$\mathcal{C}_{i,j}$	Set of all possible configurations of receiver $R_{i,j}$
$f_{i,j}$	Single interval that describes all frequencies which can be monitored by receiver $R_{i,j}$
$\mathbf{RT}$	Collection of system's receiver types
$\mathrm{RT}(R_{i,j})$	Receiver type of receiver $R_{i,j}$
RT_i	i-th system's receiver type, $\mathrm{RT}_i \in \mathbf{RT}$
$\mathcal{C}(\mathrm{RT}_i)$	Set of all possible configurations of receiver type RT_i
$f(\mathrm{RT}_i)$	Single interval that describes all frequencies which can be monitored by receiver type RT_i

Requests

$\mathcal{R}$	Set of all requests
n	Number of all requests
$\mathcal{R}_{\text{track}}$	Set of track requests
n_{track}	Number of track requests
r_{track}	Track request
$\mathcal{E}(r_{\text{track}})$	Set of emitters which belong to track request r_{track}
e	Emitter
$f(e)$	Single interval that describes at which frequencies emitter e broadcasts
$b_{\max}(e)$	Upper bound on the size of receiver configurations that can measure emitter e
$P(b_{\max})$	Probability that the randomly generated emitters have finite $b_{\max}(e)$
$\widehat{M}(r_{\text{track}})$	Maximal measurement frequency of track request r_{track}
$\widehat{m}(r_{\text{track}})$	Goal measurement frequency of track request r_{track}
$IG(r_{\text{track}})$	Information gain function of track request r_{track}
$IG'(r_{\text{track}})$	Predicted future information gain of track request r_{track}
$IG^-(r_{\text{track}})$	Last change in information gain of track request r_{track} when it was measured
$IG^+(r_{\text{track}})$	Last change in information gain of track request r_{track} when it was not measured
$\mathcal{R}_{\text{survey}}$	Set of survey requests
n_{survey}	Number of survey requests
r_{survey}	Survey request
$f(r_{\text{survey}})$	Single interval that describes which frequencies should be surveyed according to the request r_{survey}
$\widehat{m}(r_{\text{survey}})$	Goal measurement frequency of survey request r_{survey}

$\mathcal{S}(r_{\text{survey}})$	Set of sub-surveys of survey request r_{survey}
sr	Sub-survey (request)
$f(sr)$	Single interval that describes which frequencies belong to sub-survey sr
$\widehat{m}(sr)$	Goal measurement frequency of sub-survey sr
$m_{\text{hist}}(sr)$	Historical measurement frequency of sub-survey sr
$\mathcal{R}_{\text{user}}$	Set of user requests
n_{user}	Number of user requests
r_{user}	User request
$C(r_{\text{user}})$	Configuration of user request r_{user}
$SN(r_{\text{user}})$	Set of sensor nodes of user request r_{user}
$\underline{\tau}(r_{\text{user}})$	Starting measurement interval of user request r_{user}
$\overline{\tau}(r_{\text{user}})$	Ending measurement interval of user request r_{user}
$\mathcal{R}_{\text{noise}}$	Set of noise requests
n_{noise}	Number of noise requests
r_{noise}	Noise request
$f(r_{\text{noise}})$	Single interval that describes which frequencies cannot be observed according to the request r_{noise}

Construction of Tuning Plan

l	Size of the tuning plan, its duration in measurement intervals
L	Tuning plan
$\mathcal{H}$	Collection of previous tuning plans
s	Number of prediction steps during the computation of track measurement frequencies
s^-	Number of prediction steps during which the target is measured
s^+	Number of prediction steps during which the target is not measured
$\widehat{IG}$	Goal information gain for each target during the computation of track measurement frequencies
$\mathcal{B}$	Set of all blocks
B	Block
$C(B)$	Configuration of block B
$RT(B)$	Receiver type of block B
$SNN(B)$	Sensor node need of block B
$\widehat{m}(B)$	Goal measurement frequency of block B

$m(L, B)$	Realized measurement frequency of block B by tuning plan L
γ	Discount factor
$\mathcal{P}$	Set of positions
P	Position

Multiple-Interval Containment Problem

C	Set of covers
c	Multiple-interval cover
$cost(c)$	Cost of cover c
$type(c)$	Type of cover c
$\mathcal{T}_{\text{cover}}$	Set of all cover types
T	Set of targets
t	Multiple-interval target
$demand(t)$	Demand of target t
$size(t')$	Cover size limit of single interval t' from target t
$type(t)$	Type of target t
$\mathcal{T}_{\text{target}}$	Set of all target types

Miscellaneous Abbreviations

BCRLB	Bayesian Cramér-Rao lower bound
BD-AI	Bayesian detector and amplitude information
DAG	Directed acyclic graph
PC-CRLB	Predicted conditional Cramér-Rao lower bound
PCRLB	Posterior Cramér-Rao lower bound
TU	Totally unimodular

Chapter 1

Introduction

The *Passive ESM Tracker VERA-NG* is a passive radar used for detection, localization, tracking, and identification of air, ground, and naval objects traditionally called targets. To achieve all this, the system must use its receivers to observe the frequencies of interest, where targets might broadcast. Unfortunately, the total size of simultaneously monitored frequencies is relatively small due to receiver limitations and their extremely high price. Therefore, it is essential to configure them so that as many of the demanded frequencies as possible are observed and all the system-specific constraints hold. The receiver configurations during different time intervals are determined by a schedule called a tuning plan. The goal of this thesis is to propose an algorithm for the construction of optimized tuning plans. Throughout the system's runtime, frequencies of interest and constraints frequently change, hence tuning plans cannot be reused and the following tuning plan must be created during the execution of the previous one. Consequently, the proposed algorithm must be fast enough to accomplish this.

Practically all existing literature about the optimization of radar systems focuses on regular non-passive radars whose characteristics are, unfortunately, significantly different from passive radars. One of the few exceptions is the paper by Kulmon et al. [1] which even considered the same system as we. The main contribution of this work to the field of passive radars is the optimization of receiver configurations via the newly introduced multiple-interval containment problem (MICntP). This problem could be described as a well-known set cover problem (SCP) over interval-like subsets which in our case represent monitored frequencies.

The thesis is divided into six chapters if the current one is included. The second chapter contains a brief introduction to radar principles and the VERA-NG system. We also provide a literature overview of current research in the field of radar systems optimization. Then the SCP, closely connected to radar optimization, is formulated with its solution methods. We also present the existing work discussing the SCP and other related problems over interval-like objects.

In the subsequent chapter, we first define the terminology used to describe intervals which significantly simplifies the text. Then, the chapter describes a theoretical model of a passive radar system that is considered during the algorithm's construction. It also introduces four request types using which the user and radar subsystems communicate their demands to the algorithm and are later used to formulate the criterion.

The fourth chapter finally presents the proposed algorithm for the construction of tuning plans which is separated into five steps. Each of these steps is described in detail in a separate subchapter. A standalone chapter is also dedicated to the MICntP where the problem is introduced together with various constraints influencing its character and difficulty. We then derive the complexity of differently constrained MICntPs and propose methods for solving them.

In the fifth chapter, we analyze the dependence between input requests and the algorithm's run time in a simplified scenario. Throughout the experiments, input requests are randomly generated and either their number or some of their characteristics is used as an independent variable.

The final chapter summarizes the work presented in this thesis, reiterates our contribution to the optimization of radar systems, and lists possible directions for future research.

CHAPTER 2

Background and State-of-the-Art

Them bats is smart. They use radar!

— David Letterman

This chapter serves as an introduction to the current research topics, methods, and problems on which we will build in the following chapters. The first subchapter provides a quick introduction to radars and radar systems which includes their basic principles, abilities, history, and applications. It then briefly focuses on a special radar type called passive radar and one of its representatives named VERA-NG. The following subchapter describes the problems in the field of radar system optimization by summarizing some of the recently published research papers. Finally, in the last chapter, we introduce the SCP, other related problems, and their contemporary solution methods. All of which are closely related to the optimization of radar systems.

2.1 Introduction to Radar

The radar can be defined as an electromagnetic sensor for the detection and localization of reflecting objects [2]. The basic work of the radar can be described in several steps. Firstly, the radar's transmitter antenna emits electromagnetic pulses in given directions. These pulses propagate through the atmosphere and some of them are intercepted by a reflecting object, which is called the target. Part of the intercepted energy is absorbed by the target and the other part is scattered in many directions including the radar's receiver antenna. The received signal is amplified and processed to distinguish between noise and target echo from which can then be computed target location and other information.

For the calculation of the target location, the radar determines the distance to the observed object according to the round-trip time of the produced electromagnetic pulse. The azimuth and elevation angle correspond to the angular direction of the radar's antenna where the magnitude of the echo signal is maximal [3]. These three values form spherical coordinates centered at the radar position which can be transformed to obtain the target's absolute location. Some radars can also obtain the target's radial velocity (speed toward the radar), by measuring the frequency shift caused by the Doppler effect, or identify its shape and type [2].

The first principles of radar were formulated and experimentally confirmed at the end of the 19th century by James Maxwell and Heinrich Hertz. At the beginning of the 20th century, several scientists proposed to use these principles for ship navigation to avoid collisions [4]. With the fast development of aviation following the First World War, British scientist Robert Watson-Watt came up with an idea to detect military aircraft using radar. In 1939 the first radar system called Chain Home was constructed along the southern and eastern coast of Great Britain, as can be seen in Figure 2.1, to provide early warning against enemy air raids. This system played an essential role in the British victory in the Battle of Britain at the beginning of the Second World War [5]. During the war, almost all of the major nations used and experimented with radars for various tasks ranging from submarine detection to bomber navigation. Around this time, the term radar was introduced by the United States Navy as an acronym derived from **RA**dio **D**etection **A**nd **R**anging [8]. In the 1950s, radars started using the Doppler effect to measure the speed of detected objects which found its use not only in military applications but also in radar speed guns, navigation, and in meteorology to compute the movement of precipitation. Another important development was the introduction of passive phased array (PESA) antennas allowing for the tracking of several objects at once due to its ability to quickly switch the radar direction. The radar development swiftly continued throughout the entire Cold War hand in hand with the development of electronics, signal filtering, data processing, and computer technology [4].

Since the introduction of radar, its most important applications and developments have been in the military [3] where it is employed in many forms for navigation, weapon control, missile guidance, enemy detection, and tracking of various objects on the sea, land, and even on the ground. Being an essential part of almost every air defense system, it is used for surveillance, target tracking, and recognition. As already mentioned, radars are used in meteorology for weather observation and to determine wind velocity and direction at different altitudes, as is shown in Figure 2.2. This is especially useful at the airfields, around the runways, to warn before sudden wind changes which pose a danger to aircraft during takeoff and landing [2]. Other types of radars stationed at the airfields are used to observe and safely control the air traffic. Ships use radars as a safety measure to detect nearby ships, locate navigation markers in bad weather conditions, and map nearby coastlines [4]. The ground-penetrating radar can detect subsurface structures in a non-intrusive way, which is especially convenient for archeological imaging and mapping of buried artifacts and objects [2].

Many radar types exist which can be categorized in many ways, we will mention some of

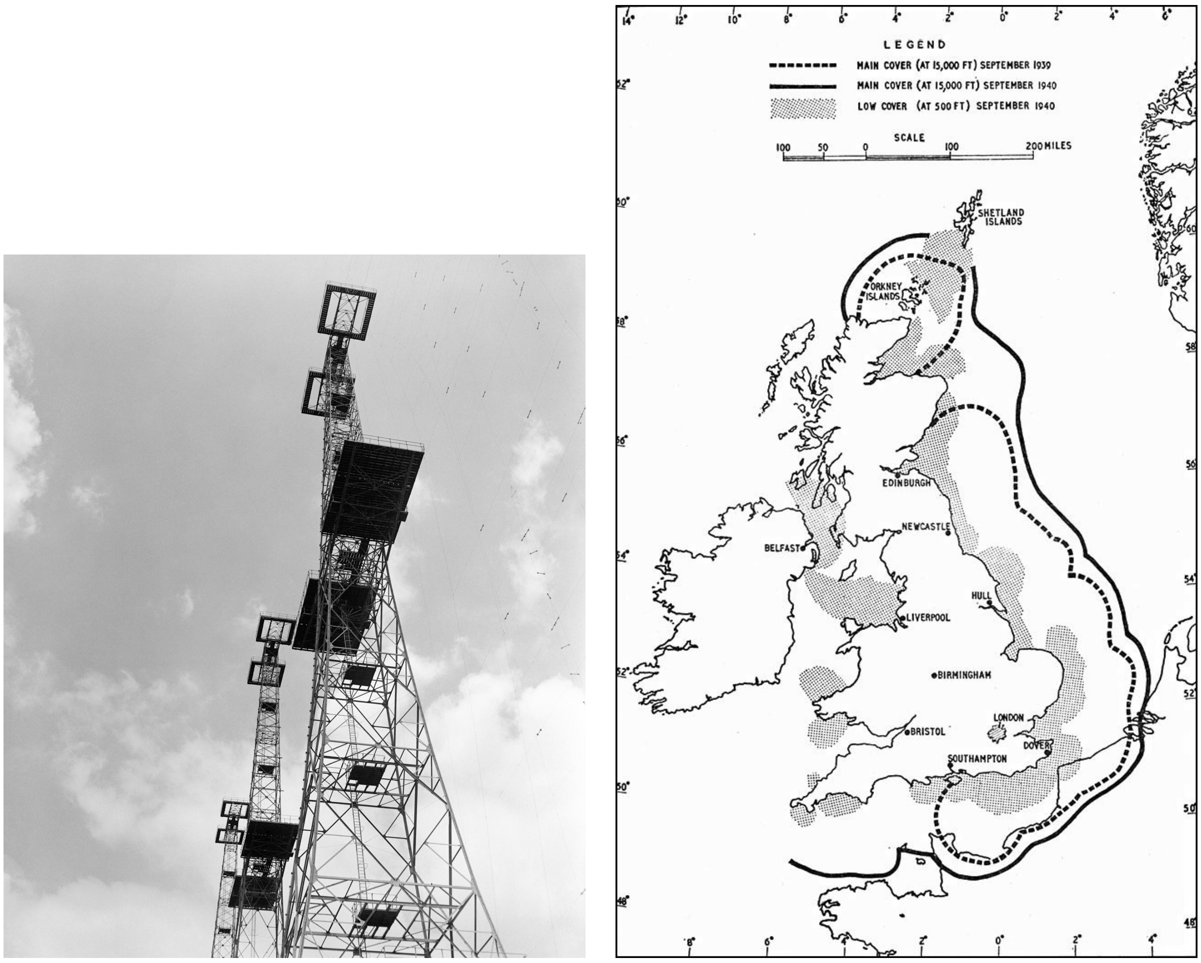


Figure 2.1: Chain Home radar system. The left image shows the transmitter towers of the Bawdsey Chain Home station [6] while the map describes the radar coverage of Great Britain at the beginning of the Second World War [7].

these categories. The radars are often described as either *monostatic* or *bistatic*. Bistatic radars have antennae for the transmitter and receiver separated by a large distance, often many kilometers. On the other hand, the antennae of monostatic radars are positioned near each other, possibly on the same structure, but more often the transmitter and receiver share a single antenna [3]. Radar's transmitter can emit two types of signal - *pulsed* or *continuous wave*. Pulsed radars switch between the transmit and receive phases which results in an emission of a repetitive series of electromagnetic pulses, by contrast, continuous wave radar emits a continuous sine-wave signal [11]. Furthermore, it is possible to group radars according to the frequencies at which they operate. Radars operating at lower frequencies are usually high-powered with large antennas which allows them to detect targets at long ranges. In comparison, high-frequency radars have higher bandwidth and produce more focused electromagnetic beams resulting in more precise localization of targets [2].

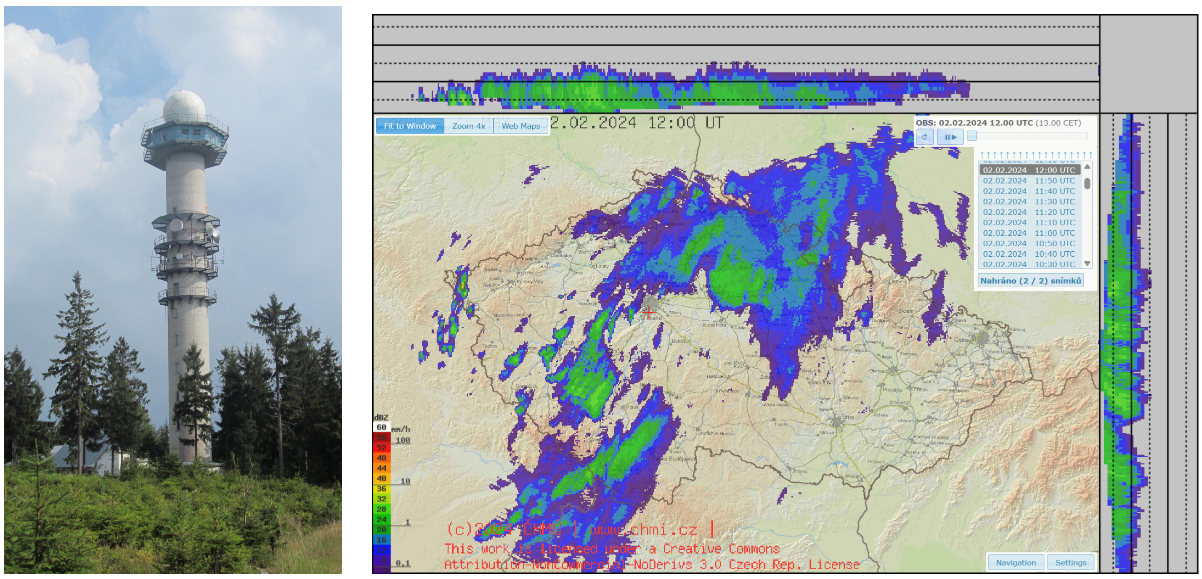


Figure 2.2: The left picture shows the weather radar on the top of Praha hill in Brdy mountains [9]. It is part of the radar network of the Czech Hydrometeorological Institute. The right figure illustrates the precipitation map produced by processing the information collected by this network [10].

Passive radar represents a unique type of radar without a transmitter that for target detection and tracking exploits signals already present in the environment. The sources of such signals are called illuminators of opportunity and can include public broadcasts, communication signals, and other radars [12]. Throughout the years many different terms for passive radar were proposed and are still used, these include: *passive coherent location* (PCL), *passive covert radar* (PCR), *parasitic radar*, and *passive surveilance*. The passive radar system usually consists of multiple receivers positioned at different locations which together with the illuminators form bistatic pairs, similar to bistatic radar, that are used for target localization. Each receiver measures the direct signal produced by the illuminator and the same signal echoed from the target, the Time Difference Of Arrival (TDOA) of those two signals is then used to compute the target's location. The difference between the two signals can be also utilized to obtain further information about the target [13]. Because the passive radar does not emit any signals, its operation is considered to be covert which is essential for its military applications.

2.1.1 VERA-NG

In its full English name, *Passive ESM Tracker VERA-NG* is a passive radar system produced by a Czech company ERA a.s. As is usual for radars, it is used for detection, localization, tracking, and identification of air, ground, and naval targets. Furthermore, as the name suggests, the system can work as an electronic support measure (ESM) that gathers, processes, and evaluates electronic intelligence (ELINT) information which can be

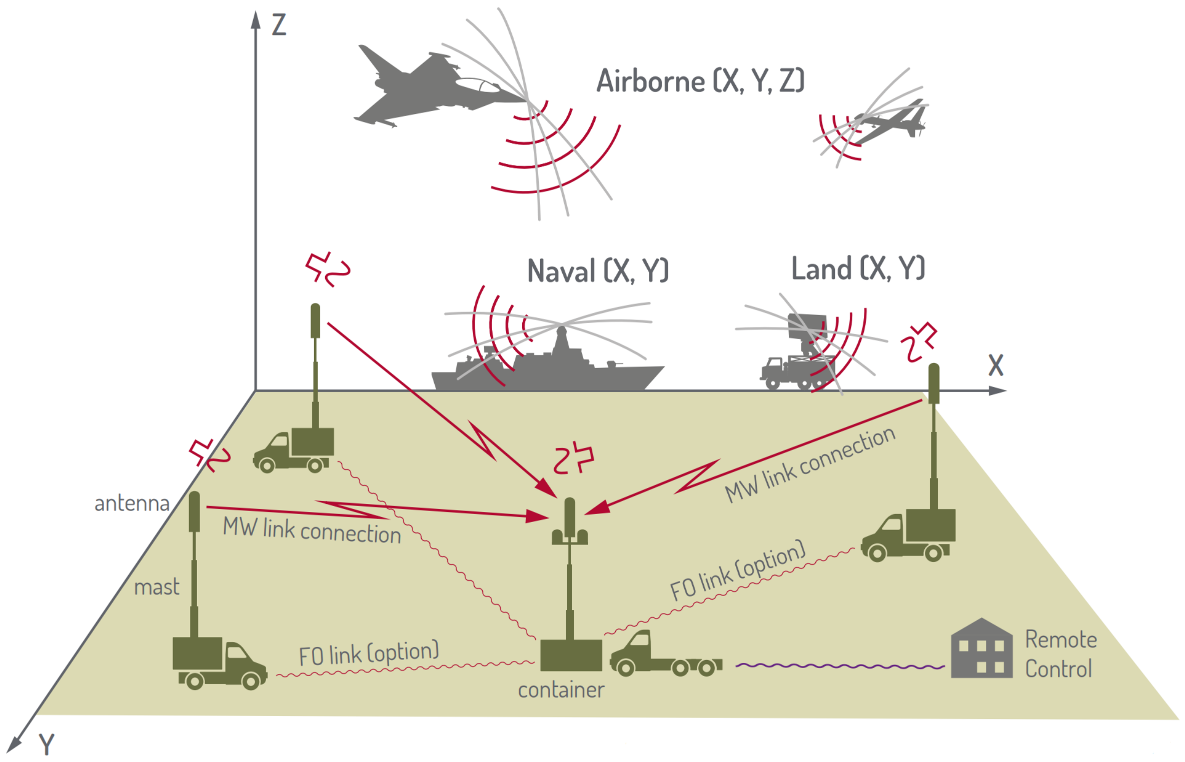


Figure 2.3: Main working principles of VERA-NG system.

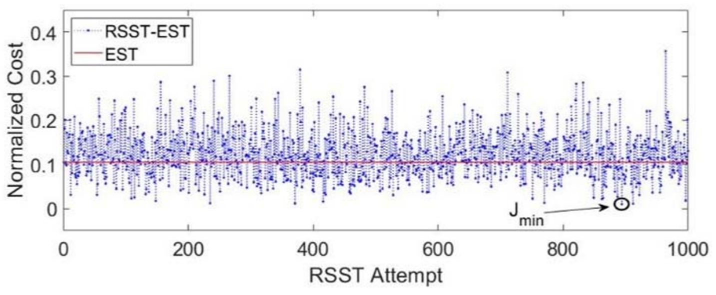
stored in a reference database of emitters for further use. The system consists of four sensor nodes, each with several receivers that can process both pulse and continuous wave signals which are then used for high-accuracy target localization based on the TDOA principle, as can be seen in Figure 2.3. One of the sensor nodes is considered to be central because it contains a central processing station that processes and evaluates all of the measured signals. The communication between the sensor nodes is provided either by microwave link connection or by fiber-optic cable.

2.2 Literature Review on Optimization of Radar Systems

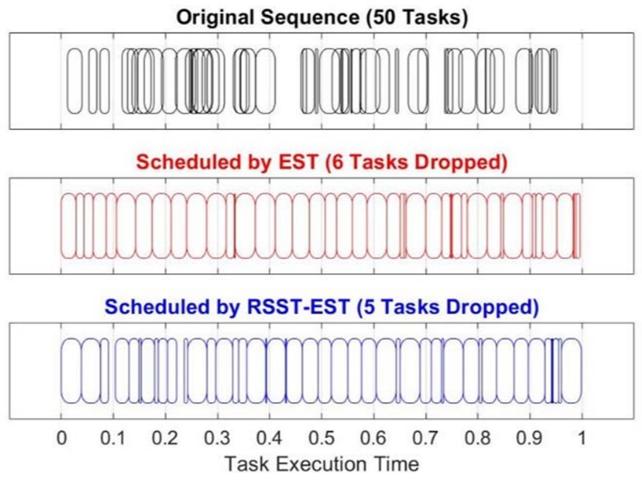
Regardless of whether it consists of a single radar or multiple transmitters and receivers, every radar system benefits from a properly optimized algorithm that can maximize its performance based on the available resources. This chapter will introduce some of the existing tasks and problems in the field of radar systems optimization and recently published methods that solve them. The optimization of individual radar behavior can include: balancing between target tracking and frequency survey, scheduling of already predetermined tasks, and the composition of various radar beam shapes. Meanwhile, the optimization of systems with multiple radars is, for example, interested in a target-radar assignment, target power allocation, and minimization of the power emitted by radars while keeping the demanded tracking accuracy. For almost all of these problems, at least some of the optimized variables are discrete, thus it is not possible to use the well-known techniques of continuous optimization and we have to approach these problems by utilizing more complex methods of combinatorial optimization. Furthermore, it has to be noted that any algorithm suitable for deployment must be able to optimize the system's behavior in real-time which is by itself problematic because the duration of tasks is of the order of tens of milliseconds.

Qu et al. [14] propose a fast radar scheduling method based on the earliest start time (EST) algorithm enhanced by Monte Carlo sampling which is called Random Shifted Start Time EST (RSST-EST). The problem formulation considers multiple non-preemptive tasks that should be scheduled into a limited time window. Each task consists of five values: the ideal start time of task execution, the dwell time - task duration, task priority, and the earliest and latest time when the task execution can start. The goal is to minimize the total cost which is the mean square error over the cost of individual tasks. When the task is scheduled between the earliest and latest time, the task cost corresponds to its normalized distance from the start time multiplied by its priority, otherwise, it is equal to its priority multiplied by the task drop penalty. At the beginning, the algorithm creates a schedule using the EST algorithm w.r.t. to ideal start times. Then the predetermined amount of Monte Carlo steps are simulated, each step consists of randomly sampling the shifted start time of each task, such that it lies between its earliest and latest time, and running EST with these shifted start times. The cheapest schedule is returned by the algorithm as a solution, the comparison of created schedules and their cost can be seen in Figure 2.4. The addition of RSST resulted in a 1.4 to 9.1 times smaller cost than the simple EST method while having low computation time requirements.

A similar problem is solved by Shaghaghi et al. [15] where they consider multiple non-preemptive tasks which are to be scheduled on several identical channels each within a given time window. Every task is determined by its duration, starting time after which it can be scheduled, deadline after which it is too late for the task to be scheduled, tardiness cost, and dropping cost. They aim to minimize the cost of a schedule which is once again equal to the sum of the costs of individual targets. The target cost corresponds to its dropping cost when the task is not scheduled, otherwise, it equals the tardiness cost multiplied by the difference between the time when it is scheduled and the start time. The proposed algorithm finds the optimal schedule using the Branch & Bound (B&B) method which constructs the optimal sequence of tasks that can be deterministically transformed into a schedule. Then the found solutions are compared with the results provided by the various heuristic algorithms based on the EST and earliest deadline (ED) principles. The authors are aware that the B&B search is too time-consuming for any possible real-world application. This problem is addressed in the follow-up paper [16] where the obtained optimal schedules are used as a dataset to train the neural network for evaluation of B&B search tree nodes and consequently for its pruning which results in a significant computational speed up at the price of solution optimality. Nevertheless, the produced solutions are close to optimum



(a)



(b)

Figure 2.4: The example of task scheduling by EST and RSST-EST methods. It consists of 50 tasks and 1000 Monte Carlo simulation steps. (a) The comparison of the normalized costs of the EST schedule and all schedules generated by RSST-EST. (b) Original sequence of tasks (black), the schedule produced by EST (red), the best schedule produced by RSST-EST (blue). [14]

and still significantly better than those produced by heuristic methods.

Briheche et al. [17] study a completely different issue - the theoretical complexity of grid cover problems in radar applications. The goal is to cover the surveillance space discretized into a grid by various rectangular beams that can be formed by radar while minimizing the cost of used beams. Therefore, this problem can be formulated as a weighted geometric set cover problem. The paper considers different radar models and splits them according to the character of their surveillance grid into three groups: line cover problems, circle cover problems, and rectangular grid cover problems. This is shown in Figure 2.5. The optimal solution of the unweighted line cover problem can be quickly found by a greedy algorithm while the unweighted version is shown to be solvable in polynomial time by dynamic programming or eventually by linear programming (LP). A similar dynamic programming method can be used to solve the circle cover problem while the authors provide

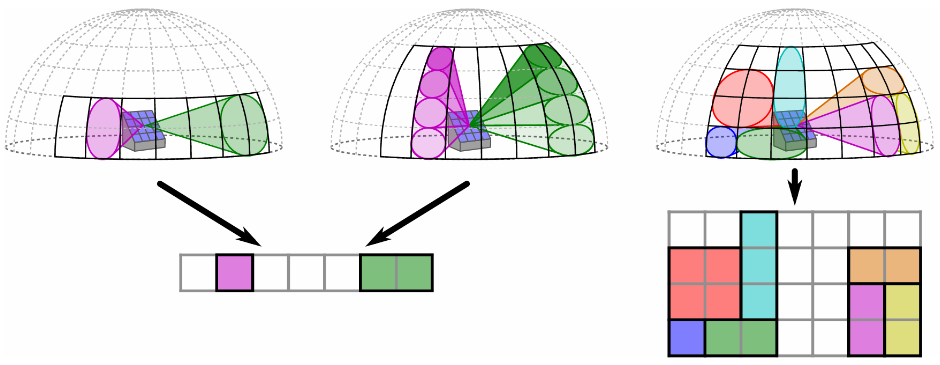


Figure 2.5: Different radar types and how they translate into the grid cover problems according to the beams they can form. Furthermore, the unidimensional cover problem (lower left) splits into the line cover problem and circle cover problem depending on whether the radar has full or limited azimuthal range. [17]

a counterexample demonstrating that LP cannot find the solution. On the other hand, it is shown that the dynamic programming approach to the rectangular grid cover problem has exponential time complexity and thus it is impractical to use. Furthermore, the vertex cover is proven to be polynomially reducible to this problem, and as a consequence, the rectangular grid cover is said to be $\mathcal{NP}$-hard and log-$\mathcal{APX}$-complete in most cases.

The above-mentioned optimization problems and their solutions are very similar to the traditional methods present in the combinatorial optimization literature, specifically in scheduling and SCP. In contrast, the following papers approach the optimization of radar systems from a different angle. They skip the abstraction layer based on tasks and directly optimize the schedule according to the expected measurement results. Consequently, they put greater detail on the proper formulation of the criterion functions and constraints that directly correspond to the target behavior and estimated outcomes of scheduled measurements.

From our point of view, Kulmon et al. [1] published a very interesting paper because it is concerned with scheduling for the same system as our work - VERA-NG. Their goal is to produce an optimized schedule that successfully maximizes and balances the tracking of existing targets (track) and the survey of frequencies for new targets (survey). The track criterion corresponds to the approximation of the expected value of the information gained when the targets are measured. The survey is equal to the Kullback-Leibler divergence of the estimated stationary distribution of surveyed frequency spectra by the constructed schedule and the stationary distribution of the optimal discrete-time Markov chain. For experiments, the paper uses the simplified problem model with a single receiver and limited-time schedule with equidistant time steps during which one of the predetermined frequency bands is tuned. First, the authors inspect the schedules produced by the NSGA-II genetic algorithm [18] for which the problem is formulated as multi-criteria. They conclude that the track and survey criteria conflict - the track prioritizes bands with targets while the survey attempts to reproduce the desired distribution. Then the problem is redefined as ϵ-constrained where the survey is minimized while the track is constrained. This problem is solved using a genetic algorithm (GA) similar to the previous one. Finally, two different rolling horizon techniques are compared, one with full and the other with partial control period. The rolling horizon is used when it is not possible to construct one schedule because the executed commands change the future schedules and therefore the schedules must be constructed repeatedly. The full control period means that the entire constructed schedule is executed. In contrast, the partial control period signifies that only a part of the schedule is carried out before the next construction. It is said that the partial control period is more stable and provides significantly better results at the cost of time-consuming computation.

Sun et al. [19] focus on multi-target tracking (MTT) in multi-radar systems (MRS) with imperfect detection and propose an algorithm called joint radar assignment and power scheduling (JRAPS). The problem model consists of several monostatic radars spread around the surveillance area, and different amounts of targets at various time intervals, with each target having its tracking accuracy requirement. Furthermore, every radar can track multiple targets at once with different power allocations that influence measurement accuracy, as can be seen in Figure 2.6. The authors derive the posterior Cramér-Rao lower bound (PCRLB) with imperfect detection and use it as a track performance metric. Next, they formulate the optimization problem with a criterion based on this metric and constraints corresponding to the system's properties. Optimized variables are the binary target-to-radar assignment matrix and real-valued power allocation matrix. The three-step algorithm is proposed to solve the aforementioned task. During the first step, the power is uniformly allocated and the target-to-radar matrix is filled by a greedy optimization strategy. In the following step, the target-to-radar matrix is fixed while the power allocation is optimized by the Zoutendijk method of feasible directions (ZMFD). In the end, the iterative elimination of target-to-radar assignments based on their global performance is done which allows for power relocation and further decreases the criteria value. The proposed metric and algorithm behavior are examined in a simulated environment with 10 radars and 3 targets at each time interval. The JRAPS method is compared with the exhaustive search-based strategy which shows that its performance is close to the optimum while being significantly faster.

Zhang et al. [20] address almost the same problem as Sun et al. [19] whose work is already described in the previous paragraph. They consider a phased array radar network (PARN) with several monostatic radars that track multiple targets. The tracker performance is measured by newly formulated predicted conditional Cramér-Rao lower bound (PC-CRLB) with Bayesian detector and amplitude information (BD-AI). The problem criterion is formulated as the minimization of the mean squared error of the difference between track metric and demanded accuracy subject to the limited number of false alarms and other system limitations. The optimization is achieved by setting the amount of time the radar

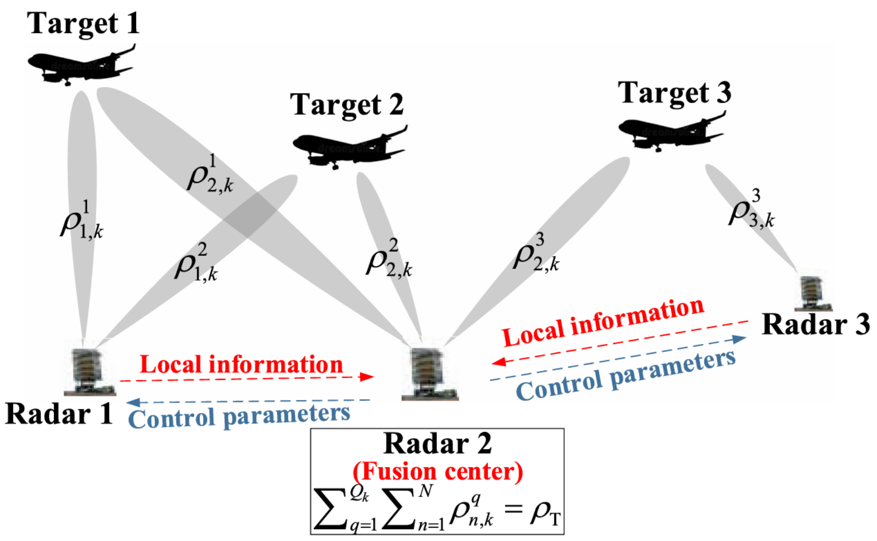


Figure 2.6: The example of MRS for MTT with a specific target-to-radar assignment and parametrized power allocations ρ that is considered in [19].

focuses on a given target (dwell time) and pre-detection thresholds which influence the false alarm count. The authors propose the cyclic two-step algorithm, the first step optimizes the dwell time, while the second step optimizes thresholds. Both optimization steps are done by the ZMFD method combined with the local search to increase the accuracy of the solution. These two steps are repeated while the improvement remains significant. The numerical simulation of PARN with 3 radars and 3 targets is done for various scenarios that confirm the proposed method's robustness.

Once again, Shi et al. [21] study the radar network with several radars which are supposed to track multiple low probability of intercept (LPI) targets. The tracking accuracy is predicted by the Bayesian Cramér-Rao lower bound (BCRLB). The system has three types of variables: binary target-to-radar selection variables that assign which targets are tracked by which radar, transmit power variables representing how much power the radar allocates for a given target, and signal bandwidth variables that influence the bandwidth of waves emitted by the radar toward the target. The problem is formulated as a minimization of total power consumption under the constraints that the target track accuracy must be maintained, each target can be tracked by only one radar, and existing system limitations. The proposed method for solving this problem checks all possible combinations of target-to-radar assignments and for each of these, the sub-problem with fixed assignment is optimized by a GA. The testing scenarios consist of a radar network with 6 monostatic radars and 2 targets, the comparison is made between the proposed method with and without the bandwidth optimization which is said to verify the correctness of the proposed methods.

2.3 Set Cover Problem and Its Variants

The SCP is a well-known optimization problem mostly studied in the fields of computer science, combinatorial optimization, and operations research. It can be used to solve many problems encountered during the optimization of radar systems, some of which were already mentioned in the previous chapter. These problems include not only the space coverage by radar beams as studied by Briheche et al. [17]. For example, the radar-to-target assignment problem could be resolved by generating a large set of feasible assignments with costs and selecting the optimal subset using the weighted SCP. The optimization version of SCP is defined as follows:

Definition 2.1. Set cover problem is denoted as $(\mathcal{U}, \mathcal{S})$ where $\mathcal{U}$ is finite set of elements $\{1, \ldots, m\}$ called the universe and $\mathcal{S}$ is a set of n subsets of $\mathcal{U}$. The problem is to find a minimum-sized subset $\mathcal{S}' \subseteq \mathcal{S}$ such that it covers the whole universe $\mathcal{U} = \bigcup_{S \in \mathcal{S}'} S$. This can be formulated as:

$$\begin{aligned} \min_{\mathcal{S}'} \quad & |\mathcal{S}'| \\ \text{s.t.:} \quad & \mathcal{U} = \bigcup_{S \in \mathcal{S}'} S \\ & \mathcal{S}' \subseteq \mathcal{S}. \end{aligned} \tag{2.1}$$

This basic formulation is often extended by the addition of subset costs which changes the criterion to the minimization of the total cost of selected subsets, such a problem is called weighted SCP (WSCP).

Definition 2.2. Weighted set cover problem is described by $(\mathcal{U}, \mathcal{S}, c)$ where $\mathcal{U}$, $\mathcal{S}$ are the same as in Definition 2.1 and function $c : \mathcal{S} \to \mathbb{R}_{\geq 0}$ assigns cost to each subset. The goal is to find a subset $\mathcal{S}' \subseteq \mathcal{S}$ such that it covers the whole universe $\mathcal{U} = \bigcup_{S \in \mathcal{S}'} S$ and its cost $\sum_{S \in \mathcal{S}'} c(S)$ is minimal. It can be written as:

$$\begin{aligned} \min_{\mathcal{S}'} \quad & \sum_{S \in \mathcal{S}'} c(S) \\ \text{s.t.:} \quad & \mathcal{U} = \bigcup_{S \in \mathcal{S}'} S \\ & \mathcal{S}' \subseteq \mathcal{S}. \end{aligned} \tag{2.2}$$

As can be seen from its formulation, the problem is very general and as a consequence, it is difficult to develop a reasonably fast optimization algorithm that works well for all possible input values. This is supported by the fact that the decision version of SCP, whose goal is to decide whether it is possible to cover the set $\mathcal{U}$ by at most k sets from $\mathcal{S}$, was proved to be $\mathcal{NP}$-complete problem by Richard Karp in 1972 and it belongs among the oldest problems to be proved so [22]. Furthermore, the optimization version of SCP as formulated in (2.1) is known to be $\mathcal{NP}$-hard [23]. With the optimal solution being computationally intractable for larger instances, the researchers focused on approximation algorithms that provide theoretical guarantees for its solution. Chvátal [24] proved that the criterion value of solution produced by the greedy approximation algorithm, which works by iteratively adding a subset $S \in \mathcal{S}$ to a solution $\mathcal{S}'$ that contains the greatest number of still uncovered elements, is at most the $H(\max_{S \in \mathcal{S}} |S|)$ multiple of the optimum, where $H(k) = \sum_{i=1}^{k} \frac{1}{i} \leq \ln k + 1$. In 2014, Dinur and Steurer [25] verified that for every $\alpha > 0$, it is $\mathcal{NP}$-hard to approximate set cover to within $(1 - \alpha) \ln n$ of optimum.

Algorithms providing exact solutions of SCP and WSCP are mainly based on the integer linear programming (ILP) or variations of B&B method [26]. The ILP that corresponds to the WSCP is formulated in (2.3), where the vector $\mathbf{x} \in \{0,1\}^m$ denotes which subsets from $\mathcal{S}$ are selected, the vector $\mathbf{c} \in \mathbb{R}^m_{\geq 0}$ corresponds to the subset costs, and i-th column of matrix $\mathbf{A} \in \{0,1\}^{m \times n}$ represents which elements are present in the i-th subset of $\mathcal{S}$. Nowadays, even relatively large SCPs can be solved in a reasonable time by both commercial and open-source general-purpose ILP solvers, like for example Gurobi, CPLEX, CBC, GLPK, or HiGHS.

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T \mathbf{x} \\ \text{s.t.:} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{1} \\ & \mathbf{x} \in \{0,1\}^m \end{aligned} \tag{2.3}$$

Since the ILP is $\mathcal{NP}$-hard and likely impossible to solve in polynomial time, it would be convenient if at least a subset of this problem was proven to be polynomial. Fortunately, it is a well-known fact [27] that:

Theorem 2.3. Any LP denoted as $\{\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$, where the matrix $\mathbf{A}$ is totally unimodular (TU) and the vector $\mathbf{b}$ is integral, has integral optima.

Corollary 2.4. When the matrix of WSCP is TU, the LP relaxation described in (2.4) has an integral solution and the whole problem can be solved in polynomial time.

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T \mathbf{x} \\ \text{s.t.:} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{1} \\ & \mathbf{x} \in [0,1]^m \end{aligned} \tag{2.4}$$

Furthermore, Schöbel [28] states that when the WSCP matrix is TU, it is possible to reformulate the problem as a network flow problem in an acyclic network. They also prove that under the same condition, the task can be transformed into the shortest path problem in a directed acyclic graph.

The methods based on the B&B principles explore a tree of all possible solutions while its branches are pruned according to the lower bounds of partial solutions. These lower bounds are usually based on the optimum value of LP relaxation of SCP. Even the computation of the relaxed problem is too time-consuming considering that it has to be done in every tree node. Consequently, Lagrangian relaxation combined with subgradient optimization is often used to find near-optimal solutions [26].

Heuristic algorithms mostly work based on a similar principle as the greedy approximation algorithm studied by Chvátal [24] - they iteratively build the solution by adding new subsets that are selected according to some heuristic function. These functions are often based on the Lagrangian costs of subsets which can be considered as by-products of solving the Lagrangian relaxation of the problem. Unfortunately, these costs must be part of a more complex heuristic because the solutions produced by different near-optimal Lagrangian costs frequently have very different criterion values [26].

The SCP can be used to formulate and solve a wide range of problems, we include some of them as examples. Since the 1970s, it has been used for various job and crew scheduling problems mostly by transportation companies. This includes the airline crew scheduling that usually consists of generating a huge amount of crew schedules according to the predefined constraints and then solving the SCP to cover all of the planned flights by the least amount of generated schedules [29][30]. Another famous application is a selection of optimal facility locations to minimize the number of them needed to cover all of the customers [31]. The facilities can include radar installations, schools, hospitals, police, and fire stations [32]. Hochbaum et al. [33] formulate the task of labor scheduling, where at each time interval the specific number of workers must be at work, as a special case of WSCP called the multicover problem (MC) defined as follows.

Definition 2.5. Multicover problem given by $(\mathcal{U}, \mathcal{S}, c, d)$, where $\mathcal{U}, \mathcal{S}, c$ have the same meaning as in Definition 2.2 and function $d : \mathcal{U} \rightarrow \mathbb{N}$ represents demand of each element. The task is to find a collection of subsets $\mathcal{S}'$ (subsets can repeat) sampled from $\mathcal{S}$ such that every element $e \in \mathcal{U}$ is present in at least $d(e)$ selected subsets and its cost $\sum_{S \in \mathcal{S}'} c(S)$ is minimal.

In (2.5) the MC problem is formulated as ILP, which is similar to the formulation of WSCP in Equation (2.3) with the difference that the vector $\mathbf{x}$ represents how many times the subset from $\mathcal{S}$ is selected and vector $\mathbf{d}$ denotes the demand of individual elements.

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.:} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{d} \\ & \mathbf{x} \in \mathbb{N}^m \end{aligned} \tag{2.5}$$

Hochbaum et al. study three different types of work shifts depending on the character of binary vectors that make up the matrix $\mathbf{A}$. When the shifts are consecutive - the vectors have consecutive ones property, the problem can be solved as minimum cost network flow. The second considered shift type is called cyclical and the vectors have consecutive zeros property. They show that the problem with cyclical shifts and constant valued vector **d** is solvable in polynomial time. On the other hand, the general cyclical shift problem is claimed to be equivalent to the exact matching problem which is known to have $\mathcal{RP}$ complexity. The final examined shift type is k-multi-shift which corresponds to the vectors that consist of k consecutive blocks of ones. The authors provide a k-approximation algorithm for this problem and also prove [33] that:

Theorem 2.6. There is a polynomial reduction of the R3-SAT problem into a 2-multi-shift MC problem.

The R3-SAT problem denotes a special case of the boolean satisfiability problem (SAT) in which each variable appears exactly three times. The R3-SAT is known to be NP-hard as described in [22], problem LO1.

Corollary 2.7. For any $k \geq 2$, the k-multi-shift MC problem is $\mathcal{NP}$-hard even when the demand vector equals $\mathbf{1}$.

Due to the complexity of the basic version of SCP, the research mostly focuses on more specific problem instances that are often significantly easier to solve. SCPs in which both the covered elements and their subsets can be represented by geometric objects are called geometric set cover problems (GSCP). For example, it is known [34] that the optimal covering of an interval by its subintervals is solvable in polynomial time. Edwards et al. [34] consider a similar problem where the interval and subinterval endpoints are integral while only a fraction of the interval has to be covered. Although this problem is optimally solvable in quadratic time, they provide an approximate algorithm with lower time complexity that is useful when working with large-volume data. It is shown that for given $\epsilon > 0$ the proposed algorithm is $(1 + \epsilon)$ approximation of optimum with $O(\frac{1}{\epsilon}\min\{m + n, n \log n\})$ time complexity, where m is the size of covered interval and m is the number of subintervals.

Another $\mathcal{NP}$-hard problem that is closely related to the SCP is called the hitting set problem (HSP). We can understand it as an "inverted" version of the SCP. Instead of selecting subsets such that all elements are covered, the task is to select elements so that all subsets contain the demanded amount of these elements. It even has similar extensions based on adding demands and costs creating the demand HSP (DHSP) and weighted DHSP (WDHSP).

Definition 2.8. Demand hitting set problem is given by $(\mathcal{U}, \mathcal{S}, d)$ where $\mathcal{U}$ is finite set of n elements called the universe, $\mathcal{S}$ is a set of subsets of $\mathcal{U}$, and function $d : \mathcal{S} \to \mathbb{N}$ indicates the demand of each subset. The goal is to find the minimum-sized subset $\mathcal{H} \subseteq \mathcal{U}$ such that each subset $S \in \mathcal{S}$ contains at least $d(S)$ elements from $\mathcal{H}$. The problem can be denoted as:

$$\begin{aligned} \min_{\mathcal{H}} \quad & |\mathcal{H}| \\ \text{s.t.:} \quad & |S \cap \mathcal{H}| \geq d(S) \quad \forall S \in \mathcal{S} \\ & \mathcal{H} \subseteq \mathcal{U}. \end{aligned} \tag{2.6}$$

Definition 2.9. Weighted demand hitting set problem is given by $(\mathcal{U}, \mathcal{S}, d, c)$ where $\mathcal{U}, \mathcal{S}, d$ are the same as in Definition 2.8, and function $c : \mathcal{U} \to \mathbb{R}_{\geq 0}$ determines the cost of each element. The task is to find the cheapest subset $\mathcal{H} \subseteq \mathcal{U}$ such that each subset $S \in \mathcal{S}$ contains at least $d(S)$ elements from $\mathcal{H}$. We can formulate the problem as follows:

$$\begin{aligned} \min_{\mathcal{H}} \quad & \sum_{e \in \mathcal{H}} c(e) \\ \text{s.t.:} \quad & |S \cap \mathcal{H}| \geq d(S) \quad \forall S \in \mathcal{S} \\ & \mathcal{H} \subseteq \mathcal{U}. \end{aligned} \tag{2.7}$$

Krupa et al. [35] consider problems of DHSP and WDHSP where the elements are points on a real line and subsets are represented by intervals. They prove that the greedy algorithm solves the DHSP version of this problem in linear time. Then, WDHSP with equal demands is shown to be solvable by minimum cost flow, and consequently, the problem has polynomial time complexity. Finally, the proposed minimum cost flow is generalized to solve any "point-interval" WDHSP.

Chapter 3

Problem Statement

Our task is to design an algorithm for producing optimized schedules that determine the behavior of passive radar systems while maximizing their efficiency limited by available resources. Even though, the presented algorithm was originally intended only for the VERA-NG system, described in Section 2.1.1, we in this and the following chapters assume a generalized passive radar system which will be described shortly. But first, to simplify the text we introduce notations that will be used to further describe intervals and unions of intervals. These are inspired by Butman et al. [36] who study the problems of minimum vertex cover, minimum dominating set, and maximum clique of multiple-interval graphs. From now on:

Definition 3.1. The interval defined as $i = [a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b;\ a < b\}$ will be called **single interval**. The set of single intervals will be denoted as 1-$\mathbb{I}$.

The leftmost point a of the single interval i will be indicated as $l(i)$. Similarly, the rightmost point b can be written as $r(i)$. Finally, the size of the single interval is denoted as $|i|$ and it is equal to $b - a$.

Definition 3.2. The **t-interval** i is the union of $t \in \mathbb{N}$ pairwise disjoint single intervals $i_1, \ldots, i_t$, such that $i = \bigcup_{k=1}^{t} i_k$. The set of t'-intervals, where $t' \leq t$, is written as t-$\mathbb{I}$.

Definition 3.3. Multiple-interval can be an arbitrary single interval or t-interval where $t \in \mathbb{N}$. The set of multiple-intervals is symbolized as $\mathbb{I}$.

Definition 3.4. We say that the multiple-interval t is **contained** by multiple-interval c or that c **contains** t if $t \subseteq c$.

These definitions are illustrated in Figure 3.1

The assumed passive radar system consists of four sensor nodes deployed at different locations, denoted as $\mathbf{SN} = (SN_0, SN_1, SN_2, SN_3)$. Each sensor node SN_i has receivers

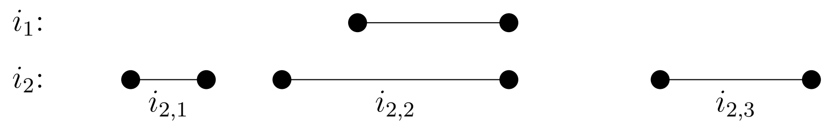


Figure 3.1: The figure shows an example of a single interval $i_1 \in$ 1-$\mathbb{I}$ and a 3-interval $i_2 \in$ 3-$\mathbb{I}$ that consists of three disjoint single intervals $i_{2,1}$, $i_{2,2}$, and $i_{2,3}$. Both i_1 and i_2 can be described as a multiple-interval. Furthermore, we can say that i_1 is contained by i_2 or that i_2 contains i_1.

$\mathbf{R}_i = (R_{i,0}, R_{i,1}, \dots)$, as can be seen in Figure 3.2a. For any receiver $R_{i,j}$, its configuration $C_{i,j}$ determines which frequencies it observes. These frequencies can be described by multiple-interval $f(C_{i,j})$. The possible configurations of a receiver are determined by its type $RT(R_{i,j})$. The collection of all receiver types present in the radar system is symbolized by $\mathbf{RT} = (RT_0, RT_1, \dots)$. Each sensor node must have at least one receiver of each type. The set of possible configurations of receivers with type RT_i is denoted as $\mathcal{C}(RT_i)$, and its union forms a single interval called the measurable frequencies $f(RT_i)$:

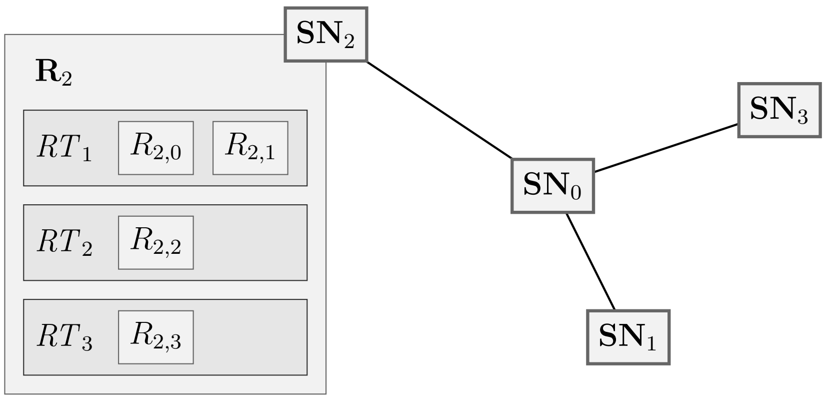
$$f(RT_i) = \bigcup_{C \in \mathcal{C}(RT_i)} f(C). \tag{3.1}$$

An example of possible measurable frequencies of different receiver types is provided in Figure 3.2b. It holds that measurable frequencies of receiver types do not overlap. To be more precise, for every pair of receiver types the size of the intersection of measurable frequencies is at most one:

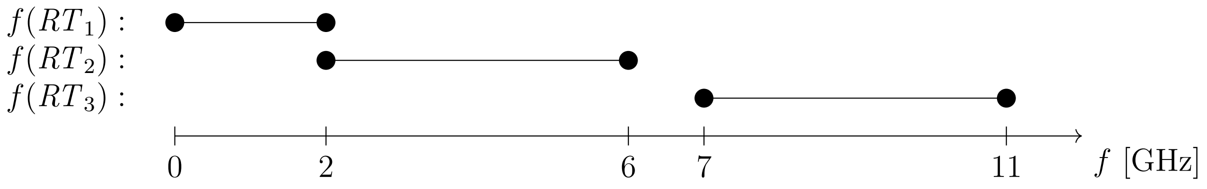
$$\forall RT_i, RT_j \in \mathbf{RT}; i \neq j : |f(RT_i) \cap f(RT_j)| \leq 1. \tag{3.2}$$

The passive radar system works in evenly sized time intervals, called measurement intervals, that are synchronized across all sensor nodes. Their assumed duration is in the lower tens of milliseconds. During each measurement interval, every receiver monitors frequencies according to its configuration. This configuration can be instantly changed at the time of transition to the next measurement interval. The schedule specifying the configuration of all receivers for l measurement intervals is called a tuning plan L of size $l = |L|$. We expect the tuning plan's size to be between 5 and 50 measurement intervals. Figure 3.2c shows an example of a tuning plan for the system presented in Figure 3.2a.

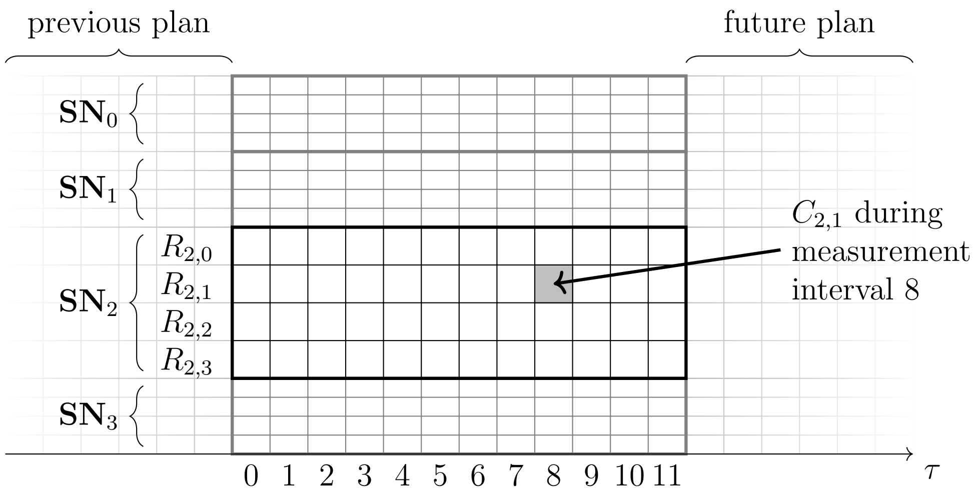
The system and user demands are communicated to the algorithm by requests. The algorithm's goal is to realize them as best as the limited resources - receivers - allow and consequently, they shape its behavior. The set of all requests is denoted as $\mathcal{R}$, its size is $n = |\mathcal{R}|$ and we assume it will be in the lower hundreds. We consider four request types: track, survey, user, and noise. It holds that $\mathcal{R} = \mathcal{R}_{\text{track}} \cup \mathcal{R}_{\text{survey}} \cup \mathcal{R}_{\text{user}} \cup \mathcal{R}_{\text{noise}}$, and consequently: $n = n_{\text{track}} + n_{\text{survey}} + n_{\text{user}} + n_{\text{noise}}$. To describe how often is a track or



(a) The Diagram shows one of the possible compositions of the assumed passive radar system. It has four sensor nodes with three receiver types. The receiver composition of the second sensor node can be seen in detail.



(b) Visualization of measurable frequencies of different receiver types in a passive radar system. It can be seen that the equation (3.2) holds.



(c) The tuning plan of length 12 for the passive radar system which has already been presented in (a). The x-axis corresponds to discrete time intervals called measurement intervals and the y-axis represents the system's receivers. Each grid tile corresponds to the planned receiver configuration at a corresponding measurement interval.

Figure 3.2: The example of an assumed Passive Radar System.

survey request realized we introduce a new unit of measurement called the measurement frequency, defined as the number of realized measurements per one measurement interval.

First, let us consider a single emitter e that is described by single interval $f(e)$ and positive real value $b_{\max}(e)$. The $f(e)$ specifies at which frequencies the emitter broadcasts while the $b_{\max}(e)$ sets the upper bound on the size of receiver configurations that can measure it. We assume that the $f(e)$ fully lies inside the measurable frequencies that belong to one of the system's receiver types. Let us consider a configuration such that at least one single interval from the configuration's multiple interval that contains $f(e)$ is lesser or equal to $b_{\max}(e)$. We say that the emitter is measured during a measurement interval τ when at least one receiver on each sensor node has this configuration during the measurement interval τ. Each *track request* $r_{\text{track}} \in \mathcal{R}_{\text{track}}$ corresponds to exactly one target tracked by the system. It is described by the set of its emitters $\mathcal{E}(r_{\text{track}}) = \{e_0, e_1, \dots\}$ and maximal measurement frequency $\widehat{M}(r_{\text{track}})$. The union of all emitter frequencies is multiple-interval denoted as $f(r_{\text{track}})$.

$$f(r_{\text{track}}) = \bigcup_{e \in \mathcal{E}(r_{\text{track}})} f(e) \tag{3.3}$$

We say that the request is realized or the target is measured when at least one of its emitters is measured as in Figure 3.3. Every measurement is likely to improve the system's information about the target. Since the request does not specify its goal measurement frequency $\widehat{m}(r_{\text{track}})$ - how often the request should be realized - it must be first computed by the algorithm. Because of this, we consider that each track request has information gain function $IG_{r_{\text{track}}}$. The function at the end of each measurement interval specifies the maximal possible amount of information gained by measuring the target in the following interval. We assume that for each track request, there exists at least one configuration that realizes this request. Otherwise, the system would not be able to measure the target.

The *survey request* $r_{\text{survey}} \in \mathcal{R}_{\text{survey}}$ is characterized by a single interval $f(r_{\text{survey}})$ and goal measure frequency $\widehat{m}(r_{\text{survey}})$. As the name suggests, it describes which frequencies $f(r_{\text{survey}})$ should be examined for potential targets and also how often $\widehat{m}(r_{\text{survey}})$ it should be done. The frequency $f \in f(r_{\text{survey}})$ is said to be surveyed during the measurement interval τ when at least one receiver has a configuration C such that $f \in f(C)$ during the interval τ. Ideally, surveys of the same frequency should be distributed on different sensor nodes throughout the time intervals to ensure complete exploration. We assume that the observed frequencies of survey requests do not overlap. More exactly, for each pair of survey requests the size of the intersection of surveyed frequencies is at most one, which is also described in the following equation:

$$\forall r_1, r_2 \in \mathcal{R}_{\text{survey}}; r_1 \neq r_2 : |f(r_1) \cap f(r_2)| \leq 1. \tag{3.4}$$

Let us consider a real-world scenario, some emitter is periodically broadcasting and a system's receiver attempts to intercept it with the same period but a different phase shift. Due to this shift, the emitter and receiver are never active simultaneously, and

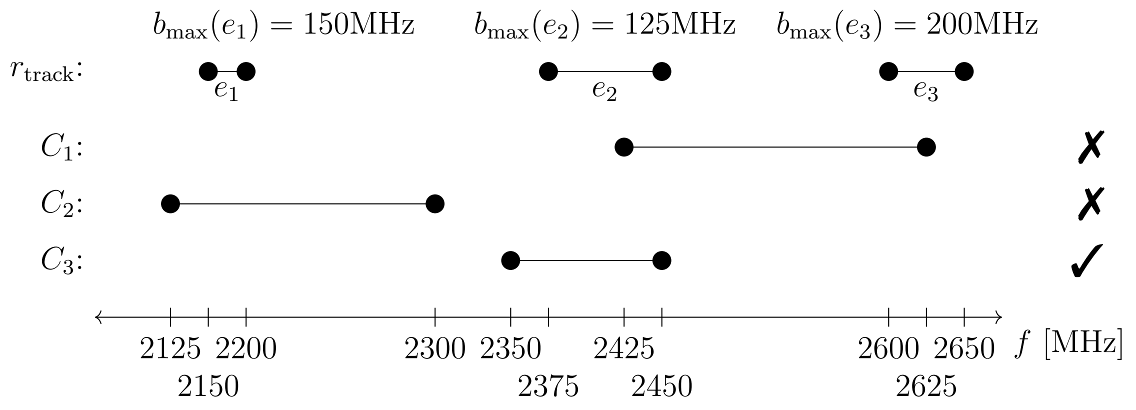


Figure 3.3: The figure shows a track request r_{track} and three sensor configurations: C_1, C_2, C_3. The request consists of three emitters $\mathcal{E}(r_{\text{track}}) = \{e_1, e_2, e_3\}$ with different maximal bandwidths. Configuration C_1 does not realize the request because it contains no emitter. Configuration C_2 does contain the emitter e_1 but it is larger than $b_{\max}(e_2)$, thus the target can not be measured by it. Finally, the third configuration C_3 contains the second emitter e_2 and is smaller than $b_{\max}(e_2)$ consequently it can realize the track request.

consequently, the receiver never intercepts the broadcasted signals. This phenomenon is called emitter-receiver synchronization and is described by Kulmon et al. [1]. To mitigate it, the realization of the same *track* or *survey* request must be scheduled aperiodically.

The *user request* $r_{\text{user}} \in \mathcal{R}_{\text{user}}$ is a high-level command that must always be scheduled. It is the only way a system or user can directly influence the tuning plan contents. Request consists of configuration $C(r_{\text{user}})$, set of sensor nodes $SN(r_{\text{user}})$, and two measurement intervals $\underline{\tau}(r_{\text{user}})$ and $\overline{\tau}(r_{\text{user}})$, such that $\underline{\tau}(r_{\text{user}}) \leq \overline{\tau}(r_{\text{user}})$. The request is scheduled if at least one receiver on each of the sensor nodes $SN(r_{\text{user}})$ has configuration $C(r_{\text{user}})$ from measurement interval $\underline{\tau}(r_{\text{user}})$ to $\overline{\tau}(r_{\text{user}})$.

The *noise request* $r_{\text{noise}} \in \mathcal{R}_{\text{noise}}$ describes at which frequencies specified by single interval $f(r_{\text{noise}})$ is present noise that negatively influences the system's performance. Therefore, these frequencies cannot be part of any scheduled configuration frequencies, except for boundary points. We also assume that the noisy frequencies are not part of any track or survey request, once again with the exception of boundary points.

To summarize, the constructed algorithm has to produce tuning plans according to the received requests. First, the algorithm has to compute the goal measurement frequency for each track request and only after that, these requests can be properly scheduled to improve the information about targets. The survey requests describe which frequencies should be scanned for new targets. It is evident that the track and survey requests at least partially contradict each other because the tracks prefer to focus on specific target frequencies while the surveys attempt to monitor as many frequencies as possible. The algorithm's goal is to create tuning plans that realize each track and survey request as many times as its goal measurement frequency while all user requests are scheduled and no frequencies specified by noise requests are observed.

Chapter 4

Methodology

Teda, to muselo dát příšernou práci, přitom taková blbost, co?

— matka Šebková, Pelíšky, 1999

The following chapter describes the proposed algorithm, also called the planner, that constructs tuning plans for passive radar systems. Since the set of requests can change during the system's runtime and the track and survey requests must be scheduled aperiodically, the tuning plan is constructed repeatedly. During the execution of one tuning plan which is expected to take hundreds of milliseconds, the algorithm must be able to create the following one. Consequently, the proposed algorithm has to be relatively fast.

It must be pointed out that collecting real-world scenarios is unrealistic because of two reasons. Firstly, all measurable frequencies would have to be monitored at all times, for which the system does not have enough receivers. Secondly, we would have to have information on every signal transmission within a radius of hundreds of kilometers which is impossible. Furthermore, creating complex artificial situations inside the simulator is extremely time-consuming. Due to these reasons, there are no existing datasets, which discourages the application of machine learning techniques on this task.

The core idea of our algorithm is that track and survey requests at similar frequencies can be realized at once by a single configuration, thus decreasing the hardware requirements and increasing efficiency. The main steps taken by the planner are outlined in Algorithm 4.1, and each of these steps is thoroughly explained in its dedicated subchapter. As already hinted in the previous chapter, the first step is to compute measurement frequencies for all track requests. This computation is based on linear extrapolation to compute the future target information gains combined with the idea that all targets should be located with the same precision. The following step splits each survey request into multiple sub-surveys which can be handled similarly to track requests. Then, it generates a large number of so-called blocks - configuration with additional information about its future insertion into a tuning plan - so that every track and sub-survey is measured by at least one block, and each block realizes as many requests as possible. We introduce the MICntP and its various extensions, for some of which we prove their computational complexity. The MICntP is used to compute the goal measurement frequencies of blocks - how often each block should be inserted into a tuning plan. The tuning plan is constructed by directly placing all user requests and then repeatedly inserting blocks to minimize the sum of squares of the difference between the goal and the realized measurement frequency of each block. These insertions are done randomly such that the produced plan is compact and the frequencies are observed aperiodically. Finally, the goal measurement frequency of sub-surveys must be updated according to the newly constructed tuning plan to prepare for future planner runs.

Algorithm 4.1: Planner

```
Input: size of tuning plan $l$, requests $\mathcal{R} = \mathcal{R}_{\text{track}} \cup \mathcal{R}_{\text{survey}} \cup \mathcal{R}_{\text{user}} \cup \mathcal{R}_{\text{noise}}$, set of
       previous tuning plans $\mathcal{H}$
Output: tuning plan $L$
1 Compute measurement frequencies for track requests $\mathcal{R}_{\text{track}}$
2 $\mathcal{B} \leftarrow$ Generate blocks according to requests $\mathcal{R}$
3 Compute measurement frequencies for blocks $\mathcal{B}$ as the MICntP
4 $L \leftarrow$ Construct tuning plan of size $l$ based on $\mathcal{B}$, $\mathcal{R}_{\text{user}}$, and $\mathcal{H}$
5 Update of sub-surveys according to $L$
```

4.1 Computing Track Measurement Frequencies

The measurement frequencies of track requests are assigned based on their predicted information gains after the execution of an ideal tuning plan that can completely realize all requests. Therefore, we must be able to predict the future information gain of each target depending on how often it is measured. Due to the complex and hard-to-predict behavior of the information gain function, we have decided on a simplistic approach based on linear extrapolation assuming that they continuously emit signal and are always successfully measured. We also expect that the plan construction is called relatively often, and thus the measurement frequencies are frequently updated limiting the error induced by this extrapolation. The future information gain $IG'(r)$ of track request $r \in \mathcal{R}_{\text{track}}$ is computed as:

$$IG'(r) := IG(r) + s^{+} IG^{+}(r) + s^{-} IG^{-}(r), \tag{4.1}$$

where $IG(r)$ is the current information gain, s^{-} describes how many times the target will be measured, and s^{+} is how many times it won't be. It holds that the number of prediction steps s equals $s^{-} + s^{+}$. Finally, $IG^{-}(r)$ and $IG^{+}(r)$ correspond to the change of the information gain the last time the target was and was not measured, respectively. An example of this extrapolation is illustrated in Figure 4.1.

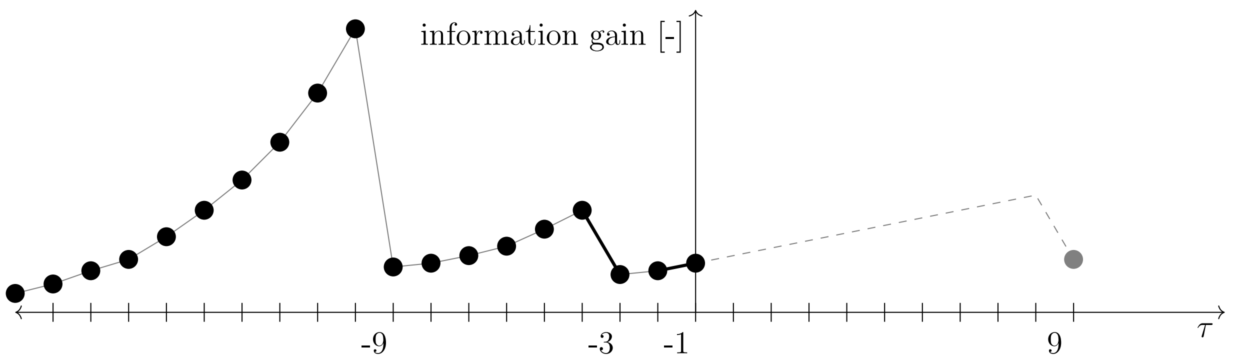


Figure 4.1: Linear extrapolation of the target's information gain based on its last changes. The x-axis represents measurement intervals. The values left of the y-axis represent the previous information gain changes while the right side shows the prediction of information gain after executing the imaginary tuning plan that once measures the target and nine times does not. The target was measured during the measurement interval -9 and -3. The $IG^-(r)$ and $IG^+(r)$ are equal to the change of the information gain during measurement interval -3 and -1, respectively.

Now, that we have described how we predict the future information gain, we can finally introduce the algorithm for computing measurement frequencies of track requests which is described by Algorithm 4.2. First, the algorithm for each track request $r \in \mathcal{R}_{\text{track}}$ predicts its information gain $IG'(r)$ as if the target was measured with its maximal measurement frequency $\widehat{M}(r)$. Then the maximum of predicted information gains is chosen as a goal information gain $\widehat{IG}$ meaning that every target's predicted information gain must be smaller or equal to this value. According to this rule, the algorithm for each target computes how many times it must be measured which is done by transforming the prediction Equation (4.1) as showed in Equation (4.2). The last step of this transformation is made under the assumption that the expression $IG^-(r) - IG^+(r)$ is negative which is reasonable to expect since the opposite would imply that the successful measurement of target decreases its tracking accuracy more than if it was not measured at all. The last step of the algorithm is to convert the measurement count to measurement frequency as can be seen at Line 10 of the algorithm.

$$\begin{aligned}
\widehat{IG} &\geq IG(r) + s^+ IG^+(r) + s^- IG^-(r) \\
\widehat{IG} &\geq IG(r) + (s - s^-) IG^+(r) + s^- IG^-(r) \\
\widehat{IG} - IG(r) - sIG^+(r) &\geq s^-(IG^-(r) - IG^+(r)) \\
s^- &\geq \frac{\widehat{IG} - IG(r) - sIG^+(r)}{IG^-(r) - IG^+(r)}
\end{aligned} \tag{4.2}$$

To summarize, the measurement frequencies of track requests are set so that their predicted information gain is lower or equal to the information gain of the worst-performing one when it is realized with its maximal measurement frequency. Consequently, this worst-performing target has a measurement frequency always set to its maximal measurement frequency.

Algorithm 4.2: Computing track measurement frequencies

```
Input: track requests $\mathcal{R}_{\text{track}}$, number of prediction steps $s$
Output: goal measurement frequency $\widehat{m}(r)\ \forall r \in \mathcal{R}_{\text{track}}$
1  $IG \leftarrow \{\}$
2  forall $r \in \mathcal{R}_{\text{track}}$ do
3      $s^- \leftarrow \lfloor s\widehat{M}(r) \rfloor$
4      $s^+ \leftarrow s - s^-$
5      $IG' \leftarrow IG(r) + s^+ IG^+(r) + s^- IG^-(r)$
6      $IG \leftarrow IG \cup \{IG'\}$
7  $\widehat{IG} \leftarrow \max\{IG\}$
8  forall $r \in \mathcal{R}_{\text{track}}$ do
9      $s^- \leftarrow \left\lceil \frac{\widehat{IG} - IG(r) - sIG^+(r)}{IG^-(r) - IG^+(r)} \right\rceil$
10     $\widehat{m}(r) \leftarrow \max\left\{0; \min\left\{\widehat{M}(r); \frac{s^-}{s}\right\}\right\}$
```

4.2 Generation of Blocks

Each survey request $r_{\text{survey}} \in \mathcal{R}_{\text{survey}}$ specifies how often the individual frequencies are to be measured but does not say that all surveyed frequencies $f(r_{\text{survey}})$ must be observed simultaneously. Because it is impossible to keep information for each surveyed frequency about how often it is measured, we have decided to split every survey request into multiple sub-surveys. This is done by dividing the surveyed frequency interval $f(r_{\text{survey}})$ from left to right into single intervals of size Δ_{survey} which are then transformed into individual sub-surveys, as can be seen in Figure 4.2. The sub-surveys set constructed from the survey request r_{survey} is denoted $\mathcal{S}(r_{\text{survey}})$. Each sub-survey $sr \in \mathcal{S}(r_{\text{survey}})$ has single interval $f(sr)$ describing its associated frequencies and goal measurement frequency $\widehat{m}(sr)$ which is initialized equal to $\widehat{m}(r_{\text{survey}})$. We say that it is surveyed or measured during the time intervals τ if at least one receiver has a configuration C such that $f(sr) \subseteq f(C)$ during the interval τ.

Thanks to the introduction of sub-surveys the realization of track and survey requests by the tuning plan is almost identical, the only difference is whether the used configuration must be present on all sensor nodes or only on a single one. And exactly for this purpose we introduce a structure called a block. The block B can be understood as a receiver configuration $C(B)$ extended by the information about receiver type $RT(B)$, sensor node need $SNN(B)$, and goal measurement frequency $\widehat{m}(B)$ which is initialized to 0.0. As the

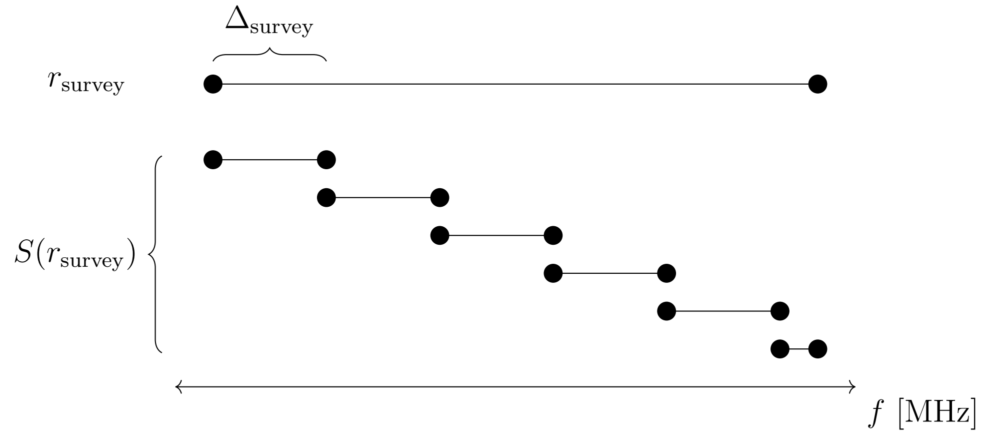
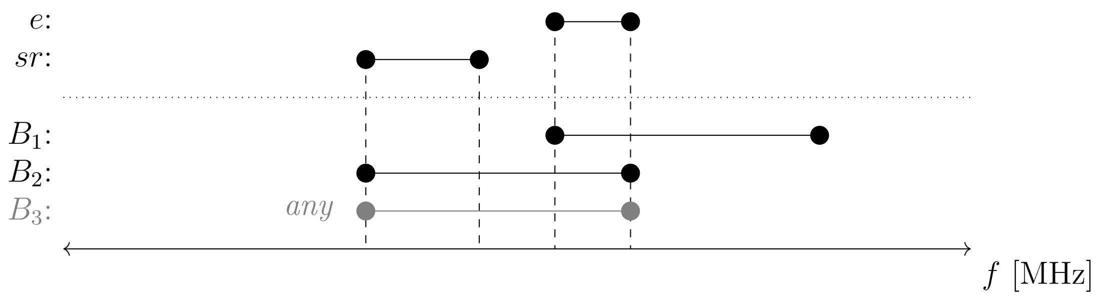


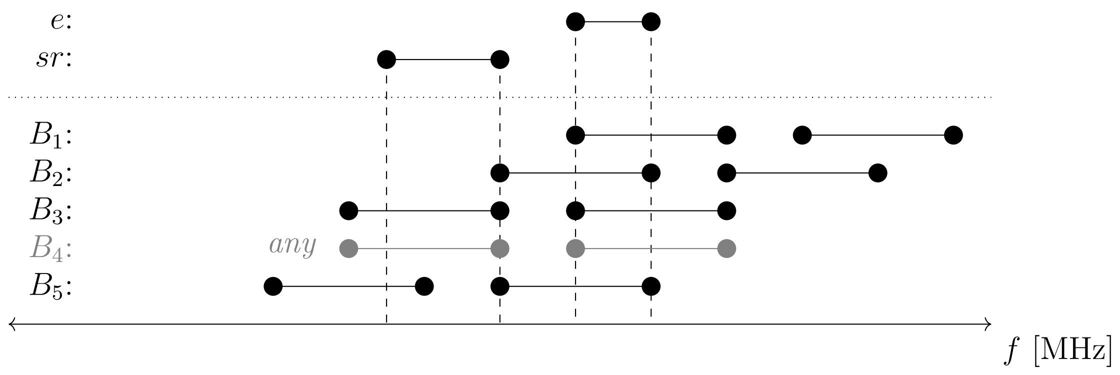
Figure 4.2: Split of a survey request r_{survey} into sub-surveys $\mathcal{S}(r_{\text{survey}})$.

name suggests, the receiver type specifies to which receiver type the block's configuration belongs. The sensor node need has two possible values *all* and *any* that determine whether the block's configuration must be set on all sensor nodes or only one arbitrary sensor node. We say that the block realizes or measures the track request or sub-survey when its insertion into a tuning plan realizes the request.

The presented algorithm generates blocks and later uses them to construct the optimized tuning plan. For this to work properly, the blocks must have the following properties: each block realizes as many requests as possible, no block monitors noisy frequencies, the track requests and sub-surveys realized by blocks are diverse, and every request and sub-survey must have at least one block that realizes them. In addition, the number of blocks must be kept reasonably low, otherwise, the algorithm may become overwhelmed in the subsequent steps. The algorithm for the generation of blocks is described in Algorithm 4.3. We generate these blocks by the heuristic, which we call left-right, based on the principle that every maximum-sized block can be uniquely identified by the left-most or right-most request it measures. In this paragraph, we will collectively refer to each emitter of every track request and every sub-survey as a *parent request.* For each parent request, the heuristic generates a group of blocks that realize the parent, their configuration has the maximum size possible and does not overlap with noisy frequencies. Thus, for each track or survey request, there exists a block that realizes it. Furthermore, the parent request must be as much to the left or right of any single interval from the block's configuration as possible. The *SNN* of each block depends on which requests its configuration can realize. When only track requests or sub-surveys can be realized the *SNN* is equal to *all* or *any*, respectively. When it is possible to realize both the tracks and sub-surveys the block is duplicated and one has the *SNN* set to *all* and the other to *any*. This is shown in Figure 4.3 which displays the generated blocks for a single parent request and two different configuration shapes. Overall, the left-right heuristic generates a diverse set of overlapping blocks with most of the possible request combinations. Some might suggest that adding both the leftmost and rightmost blocks for



(a) The maximum-size configuration is a single interval. Since block B_1 realizes only the track request, its *SNN* is set to *all*. The situation is different with block B_2 because it realizes both the track by measuring one of its emitters e and sub-survey request sr. The *SNN* of this block is *all* and its duplicate block B_3 has *SNN* equal to *any*.



(b) The maximum-size configuration is a 2-interval. Because most of the generated blocks realize only the track request, their *SNN* is set to *all*. This differs from block B_3 which measures both emitter e and sub-survey request sr. *SNN*(B_3) equals *all* and its duplicate block B_4 has *SNN* set to *any*.

Figure 4.3: Blocks generated by the left-right heuristic for the emitter e, which is part of some track request, and for two different shapes of maximum-size configuration. It can be seen that configurations of blocks are selected so that the emitter is always as much to the left or right of some configuration's single interval as possible.

each parent request is unnecessary and only the leftmost or rightmost one would suffice. Figure 4.4 is a counterexample to this statement by showing that when a block realizes a request that is not its parent, it might not be true that the blocks generated by this request must include a block that can measure the parent of the original block. Therefore both the leftmost and rightmost blocks for each parent request must be constructed.

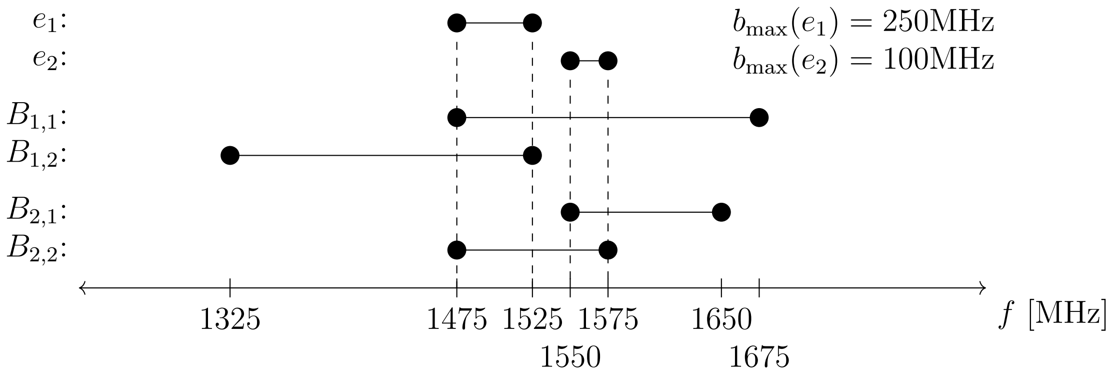


Figure 4.4: This diagram illustrates the situation when the blocks $B_{1,1}, B_{1,2}$ with parent e_1 cannot realize the emitter e_2 due to the limit on the configuration size. But the block $B_{2,2}$ generated by the parent e_2 can also realize the emitter e_1. Therefore, if we were to generate blocks whose parent is only to the left as possible, the block that can measure both emitters at once would not be constructed.

Algorithm 4.3: Generation of blocks

```
Input: track requests $\mathcal{R}_{\text{track}}$, survey requests $\mathcal{R}_{\text{survey}}$, noise requests $\mathcal{R}_{\text{noise}}$
Output: blocks $\mathcal{B}$

1 $\mathcal{B} \leftarrow \{\}$
2 forall $r_{\text{track}} \in \mathcal{R}_{\text{track}}$ do
3     forall $e \in \mathcal{E}(r_{\text{track}})$ do
4         $\mathcal{B}' \leftarrow \{B \mid B$ realizes $e$; no noise; $e$ is leftmost/rightmost of $B$; $all/any\}$
5         $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{B}'$
6 forall $r_{\text{survey}} \in \mathcal{R}_{\text{survey}}$ do
7     forall $sr \in S(r_{\text{survey}})$ do
8         $\mathcal{B}' \leftarrow \{B \mid B$ realizes $sr$; no noise; $sr$ is leftmost/rightmost of $B$; $all/any\}$
9         $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{B}'$
```

4.3 Multiple-Interval Containment Problem

This subchapter introduces a new optimization problem that is a generalization of a problem that will be encountered in the following chapter when determining block measurement frequencies. The problem is also very similar to the already-introduced SCP. We present a special notation that helps with the representation of different related problems created by adding constraints to the general problem. Finally, some possible and useful constraints are described - their definition, how they influence the problem, and sometimes even how the subproblems to which they correspond can be efficiently solved.

Definition 4.1. Multiple-interval containment problem (MICntP): given a finite set of targets T and a finite set of covers C, both consisting of multiple-intervals, find the smallest subset of covers $C' \subseteq C$ such that each target is contained by at least one cover from C'. The problem is described by the tuple (C, T) and it can be formulated as follows:

$$\begin{aligned} \min \quad & \sum_{c \in C} x_c \\ \text{s.t.:} \quad & \sum_{c \in C \colon c \text{ contains } t} x_c \geq 1 \quad \forall t \in T \\ & x_c \in \{0,1\} \quad \forall c \in C. \end{aligned} \tag{4.3}$$

The exemplary instance of the MICntP can be seen in Figure 4.5 which consists of four targets and four covers. One of the feasible solutions is a set $\{c_1, c_2, c_3\}$ because every target is contained by some of the selected covers (the c_1 contains targets t_1 and t_4, c_2 contains t_3, and c_3 contains t_2). The optimal solution is a set $\{c_1, c_4\}$ (c_1 contains t_1 and t_4, c_4 contains t_2 and t_3).

It is evident that the basic formulation of the MICntP is practically equivalent to the SCP and consequently, it might be considered uninteresting. This significantly changes when we limit the possible shape of targets and covers, from the multiple-intervals to, for example, 2-intervals, or when further constraints are added to the problem. To simplify these problem descriptions and more easily differentiate between various instances, we introduce a notation that describes problems related to the general MICntP, called families. The notation consists of two parts divided by a vertical bar and enclosed by round brackets, as can be seen in the following equation:

$$(\text{cover set } \mathbb{C}\text{; cover constraints} \mid \text{target set } \mathbb{T}\text{; target constraints}). \tag{4.4}$$

The first part describes the set, which contains all possible covers, and various cover constraints. The second part does the same for targets. For example, the general MICntP as described by Definition 4.1 can be denoted as $(\mathbb{I} \mid \mathbb{I})$. Furthermore, the problem instance visualized in Figure 4.5 belongs to the family $(2\text{-}\mathbb{I} \mid 3\text{-}\mathbb{I})$.

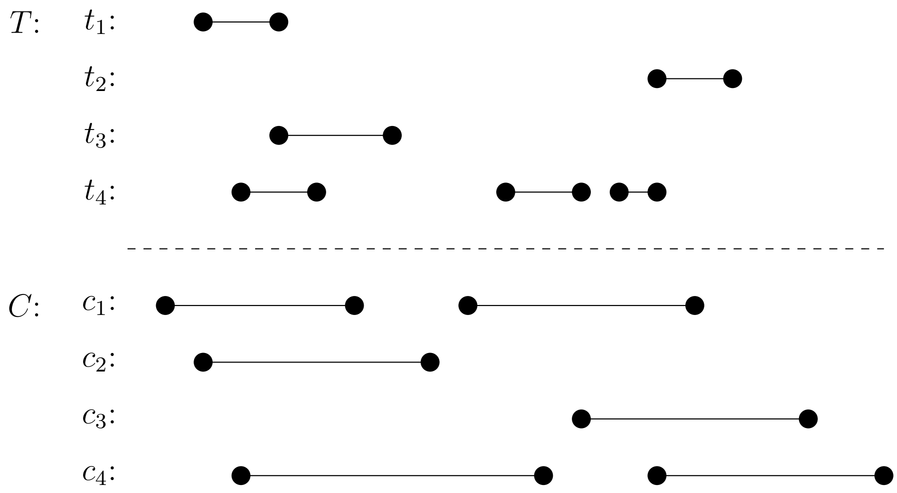


Figure 4.5: The exemplary instance of the MICntP. The optimal solution is $\{c_1, c_4\}$. This instance belongs to a family denoted as (2-$\mathbb{I}$ | 3-$\mathbb{I}$).

Corollary 4.2. A MICntP instance (C, T) belongs to a family if $C \subseteq \mathbb{C}$, all covers fulfill given cover constraints, $T \subseteq \mathbb{T}$, and all targets satisfy the target constraints.

Definition 4.3. Family $\mathcal{F}_1 = (\mathbb{C}_1;\ \textit{coverConstraints1} \mid \mathbb{T}_1;\ \textit{targetConstraints1})$ is a subset of family $\mathcal{F}_2 = (\mathbb{C}_2;\ \textit{coverConstraints2} \mid \mathbb{T}_2;\ \textit{targetConstraints2})$ denoted as $\mathcal{F}_1 \subseteq \mathcal{F}_2$ when cover and target constraints are the same, $\mathbb{C}_1 \subseteq \mathbb{C}_2$, and $\mathbb{T}_1 \subseteq \mathbb{T}_2$.

For example, (1-$\mathbb{I}$ | 1-$\mathbb{I}$) $\subseteq$ (2-$\mathbb{I}$ | 1-$\mathbb{I}$) holds true because both families do not have any constraints and 1-$\mathbb{I}$ $\subseteq$ 2-$\mathbb{I}$ according to the formulation of t-$\mathbb{I}$ in Definition 3.2.

The following chapters present several cover and target constraints that often seriously influence the problem's complexity. For some special cases of constrained problems, we either show its transformation to an existing problem and prove its correctness or we introduce the algorithm for solving such a problem.

4.3.1 No Containment

The first presented constraint is essential for solving many problems in polynomial time. This constraint can be introduced to both the cover and target intervals and it is denoted by the keyword "no containment":

$$(\mathbb{C};\ \text{no containment} \mid \mathbb{T}) \tag{4.5}$$

$$(\mathbb{C} \mid \mathbb{T};\ \text{no containment}). \tag{4.6}$$

As the name suggests, it must hold that for each interval i_1 from the interval set I, there is no other interval $i_2 \in I$ that contains it, as described in (4.7).

$$\forall i_1 \in I : \nexists i_2 \in (I \setminus \{i_1\}) : i_1 \subseteq i_2 \tag{4.7}$$

For single intervals, Equation (4.7) can be reformulated as:

$$\forall i_1 \in I : \nexists i_2 \in (I \setminus \{i_1\}) : l(i_2) \leq l(i_1) \land r(i_1) \leq r(i_2). \tag{4.8}$$

The interval set I represents either the cover set C or the target set T depending on where the constraint is present in the problem description.

Now, we introduce and prove two theorems that will be useful later for proving the correctness of the introduced methods. These theorems and their proofs are very similar but, unfortunately, different enough that they must be formulated separately.

Theorem 4.4. For a set of single intervals without containment T and arbitrary single interval c, when the set T is ordered into a sequence of single intervals such that their right endpoints are non-decreasing, the intervals from T contained by c form a consecutive subsequence.

Proof. Assume that the intervals from T are sorted so that their right endpoints are non-decreasing. This forms a sequence $(t_1, t_2, \ldots, t_{|T|})$ where:

$$\forall m, n \in \{1, \ldots |T|\}; m < n : r(t_m) \leq r(t_n). \tag{4.9}$$

Assume that the interval at index i is the left-most one which is contained by c. If no such interval exists, no intervals are contained by c, thus the consecutive subsequence is empty and the statement is true. When such interval exists, either all of the following intervals are contained by c and form a consecutive subsequence $(t_i, \ldots, t_{|T|})$ or there exists an interval at index j which is contained by c while the interval at index $j+1$ is not contained by c. Because t_{j+1} is not contained by c we have that either $r(c) < r(t_{j+1})$ or $l(t_{j+1}) < l(c)$. Let us consider that the second inequality is true. Due to Equation (4.9) we have $r(t_j) \leq r(t_{j+1})$, while $l(c) \leq l(t_j)$ holds because c_j is covered by t. The last three mentioned inequalities can be combined:

$$l(t_{j+1}) < l(c) \leq l(t_j) < r(t_j) \leq r(t_{j+1}). \tag{4.10}$$

Equation (4.10) would imply that t_{j+1} contains t_j which can not be true because of the theorem's assumption that the intervals from T do not contain each other. Therefore, $l(c) \leq l(t_j)$ cannot be true, and consequently $r(c) < r(t_{j+1})$ must always hold. Then for any t_k where $j+1 \leq k$ we have $r(c) < r(t_{j+1}) \leq r(t_k)$, as a result of the cover ordering, hence t_k cannot be contained by c. Consequently, $(t_i, \ldots, t_j)$ is a consecutive subsequence of all intervals from T contained by c. □

Theorem 4.5. For a set of single intervals without containment C and arbitrary single interval t, when the set C is ordered into a sequence of single intervals such that their right endpoints are non-decreasing, the intervals from C containing t form a consecutive subsequence.

Proof. Assume that the intervals from C are sorted so that their right endpoints are non-decreasing. This forms a sequence $(c_1, c_2, \ldots, c_{|C|})$ with the same property as described in Equation (4.9). Assume that the interval at index i is the left-most one which covers t. If no such interval exists, no intervals cover c, thus the consecutive subsequence is empty and the statement is true. When such interval exists, either all of the following intervals contain t and form a consecutive subsequence $(c_i, \ldots, c_{|C|})$ or there exists an interval at index j which contains t while the interval at index $j+1$ does not contain t. Because c_{j+1} does not contain t we have that either $l(t) < l(c_{j+1})$ or $r(c_{j+1}) < r(t)$. The second option is not possible. Due to the ordering we have $r(c_j) \leq r(c_{j+1})$. By joining the last two inequalities we get:

$$r(c_j) \leq r(c_{j+1}) < r(t), \tag{4.11}$$

this cannot be true, because it implies that the t is not covered by c_j which is a contradiction. Consequently,

$$l(t) < l(c_{j+1}) \tag{4.12}$$

must be true. Then for any c_k where $j+1 < k$ we have either $l(c_k) < l(c_{j+1})$ or $l(c_{j+1}) \leq l(c_k)$. The first inequality cannot be true because when it is combined with interval order we get:

$$l(c_k) < l(c_{j+1}) < r(c_{j+1}) \leq r(c_k), \tag{4.13}$$

implying that c_k contains c_{j+1} which is in contradiction with our initial statement. Therefore, the second inequality holds and it can be combined with Equation (4.12) resulting in $l(t) < l(c_{j+1}) \leq l(c_k)$, hence c_k cannot contain t. Consequently, $(c_i, \ldots, c_j)$ is a consecutive subsequence of all intervals from C containing t. □

4.3.2 Cover Cost

The natural extension of the general problem (4.3) is the addition of cover costs. The cost of cover c, denoted as $cost(c)$, represents how much the criteria increases when the cover is active, more precisely when $x_c = 1$. In our notation, this constraint is indicated by the keyword "cost" which belongs among the cover conditions:

$$(\mathbb{C};\ \text{cost} \mid \mathbb{T}). \tag{4.14}$$

The only change in the original problem Definition 4.1 is the addition of cover cost to the problem's input (C, cost, T) and to the criteria function:

$$\begin{aligned} \min \quad & \sum_{c \in C} cost(c)\, x_c \\ \text{s.t.:} \quad & \sum_{c \in C:\, c \text{ contains } t} x_c \geq 1 \quad \forall t \in T \\ & x_c \in \{0,1\} \quad \forall c \in C. \end{aligned} \tag{4.15}$$

For later use, we also include an ILP formulation of this problem. The vector $\mathbf{x}$ represents cover usage and matrix $\mathbf{A} \in \{0,1\}^{|T| \times |C|}$ signalizes whether the cover c contains target t. To be more precise, each matrix column $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_{|C|}]$ corresponds to one cover and its i-th value describes whether this cover contains the i-th target.

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathit{cost}(\mathbf{x})^T \mathbf{x} \\ \text{s.t.:} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{1} \\ & \mathbf{x} \in \{0,1\}^{|C|} \end{aligned} \tag{4.16}$$

Now, we formulate a theorem that will allow us to generalize future results that are based on the no containment of targets.

Theorem 4.6. Any multiple-interval containment problem with cover costs ($\mathbb{I}$; cost | $\mathbb{I}$) can be transformed without loss of generality into a similar problem without target containment described as ($\mathbb{I}$; cost | $\mathbb{I}$; no containment). The transformation algorithm takes problem $\mathcal{A} = (C, \text{cost}, T)$ and removes targets that are contained by any other target:

$$T' = \{t \in T \mid (\nexists t' \in T)[t \subset t']\}, \tag{4.17}$$

producing problem $\mathcal{B} = (C, \text{cost}, T')$.

Proof. We will show that any feasible solution of one problem is also the feasible solution to the other problem which will be done in two steps. In the first step, we show that any feasible solution to the problem $\mathcal{A}$, denoted as $C'_{\mathcal{A}}$, is also a feasible solution to problem $\mathcal{B}$. This is evident because $T' \subseteq T$. As a consequence, any solution that covers all multiple-intervals in T also covers all multiple-intervals in T'. Therefore any feasible solution to the problem $\mathcal{A}$ is also the feasible solution to problem $\mathcal{B}$.

In the second step, we show that any feasible solution to problem $\mathcal{B}$, denoted as $C'_{\mathcal{B}}$, is also a feasible solution to problem $\mathcal{A}$. To prove this, we have to show that for every target in T exists a cover from $C'_{\mathcal{B}}$ that contains it. This by definition holds for all $t' \in T'$. The remaining targets are those that were removed during the transformation $t'' \in (T \setminus T')$. From Equation (4.17) follows that every such target t'' is contained by some $t' \in T'$ which is, as already mentioned, further contained by some cover $c \in C'_{\mathcal{B}}$. Consequently, by the transitivity property of containment every t'' is contained by some cover from $C'_{\mathcal{B}}$. Hence, any feasible solution to problem $\mathcal{B}$ is also a feasible solution to problem $\mathcal{A}$.

Because any feasible solution to one problem is also feasible to the other problem and the cover costs and criteria are the same for both problems, the optimal solutions must also be the same. □

Theorem 4.7. Containment problems with cover costs, single-interval covers, and single-interval targets, denoted as (1-$\mathbb{I}$; cost | 1-$\mathbb{I}$; no containment), can be solved in polynomial time.

Proof. The stated problem can be rewritten as a relaxation of the ILP formulated in (4.16) which results in LP (4.18). Following Theorem 4.4, the targets can be sorted by their right endpoints so that the matrix $\mathbf{A}$ consists of columns with the consecutive-ones property and as a consequence, the problem matrix is TU. Therefore, according to Theorem 2.3, the LP has an integral optimum and the problem can be solved in polynomial time.

$$\begin{aligned} \min_{\mathbf{x}} \quad & cost(\mathbf{x})^T \mathbf{x} \\ \text{s.t.:} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{1} \\ & \mathbf{x} \in [0,1]^{|C|} \end{aligned} \tag{4.18}$$

□

From the total unimodularity of matrix $\mathbf{A}$, we conclude that problems which belong to (1-$\mathbb{I}$; cost | 1-$\mathbb{I}$; no containment) problem family can be reformulated and solved as a min-cost flow problem in DAG or as a shortest path problem in DAG as already mentioned in Section 2.3.

Corollary 4.8. All problems that can be denoted as (1-$\mathbb{I}$; cost | 1-$\mathbb{I}$) are solvable in polynomial time.

Theorem 4.9. The problem family described by (2-$\mathbb{I}$; cost | 1-$\mathbb{I}$) is $\mathcal{NP}$-hard.

Proof. The 2-multi-shift MC problem with a demand vector filled with ones, which will be referred to as 2MCd1 to simplify the following text, is known to be $\mathcal{NP}$-hard as stated in Corollary 2.7. The 2MCd1 directly corresponds to a subset of the family (2-$\mathbb{I}$; cost | 1-$\mathbb{I}$) of the MICntP. This can be seen when we compare the formulation of the MC problem in Equation (2.5) and the matrix formulation of the MICntP with cost constraint in Equation (4.16). The only difference is that the optimized vector in the MICntP is restricted to binary values while in the 2MCd1 problem values are natural numbers. However, because the demand vector of the 2MCd1 problem is equal to $\mathbf{1}$, there is no reason to select one shift more than once. Consequently, the optimized vector $\mathbf{x}$ can be limited to binary values without loss of generality.

Figure 4.6 illustrates the transformation from the 2MCd1 problem to the MICntP belonging to (2-$\mathbb{I}$ | 1-$\mathbb{I}$) family. Each one from the demand vector directly translates into a target whose multiple-interval has collapsed into a single value equal to the one's index in the demand vector. Similarly, every work shift represented by the vector in matrix $\mathbf{A}$ is transformed into a cover represented by a 2-interval.

Therefore, the subset of problem family (2-$\mathbb{I}$; cost | 1-$\mathbb{I}$) is $\mathcal{NP}$-hard and consequently the whole problem family must be also $\mathcal{NP}$-hard. □

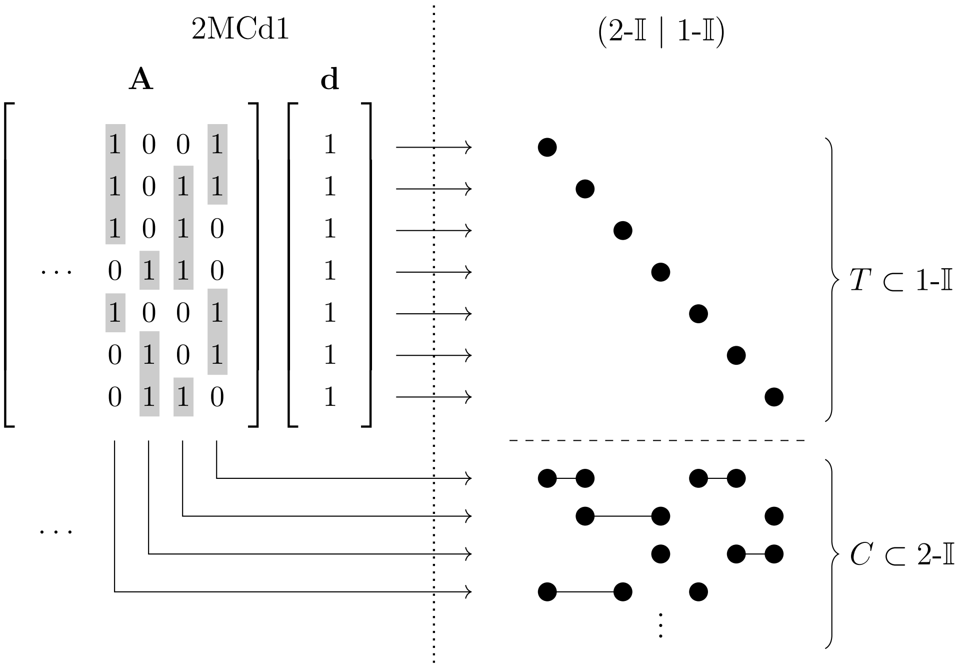


Figure 4.6: Example of direct transformation of a 2-multi-shift MC problem instance whose demand vector is filled with ones, which is denoted as 2MCd1, into a MICntP instance from the family (2-𝕀 | 1-𝕀).

Corollary 4.10. For an arbitrary $n \in \{2, 3, \dots\}$ and $m \in \{1, 2, \dots\}$, the problem family (n-𝕀; cost | m-𝕀) is $\mathcal{NP}$-hard.

4.3.3 Target Demand

In this section, we will introduce a new constraint that will significantly alter the problem formulation resulting in its notable simplification. The newly added constraint represents the notion that each target t has a demand, denoted as *demand*(t), which can be met by covers that contain the target. We denote this constraint in our problem notation by the keyword "demand" :

$$(\mathbb{C} \mid \mathbb{T};\ \text{demand}). \tag{4.19}$$

Target demand changes the original problem formula (4.3). The binary variable x_c, which indicates whether the cover is used in the original formula, becomes a positive real-valued variable representing how much of the target's demand can be met by the corresponding cover c. Furthermore, the right-hand side of the first equation changes from 1 to the target demand *demand*(t). Formulation (4.20) holds for any possible form of the covers and targets. It is also evident that such a problem can be solved in polynomial time by LP.

$$
\begin{aligned}
\min_{\forall c \in C\colon x_c} \quad & \sum_{c \in C} x_c \\
\text{s.t.:} \quad & \sum_{c \in C\colon c \text{ contains } t} x_c \geq \mathit{demand}(t) \quad \forall t \in T \\
& x_c \in \mathbb{R}_{\geq 0} \quad \forall c \in C
\end{aligned}
\tag{4.20}
$$

Corollary 4.11. Any problem with the target demand constraint, which can be formulated as ($\mathbb{I}$ | $\mathbb{I}$; demand) is solvable in polynomial time.

Theorem 4.12. Problems with the positive integer-valued target demands, single-interval covers without containment, and single-interval targets, which are in our notation described as (1-$\mathbb{I}$; no containment | 1-$\mathbb{I}$; demand $\in \mathbb{N}$), have integral optimum.

Proof. We can rewrite the LP into a matrix form:

$$
\begin{aligned}
\min_{\mathbf{x}} \quad & \mathbf{1}^T \mathbf{x} \\
\text{s.t.:} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{d} \\
& \mathbf{x} \in \mathbb{R}^{|C|}_{\geq 0}\,,
\end{aligned}
\tag{4.21}
$$

where a vector $\mathbf{x}$ represents the demands met by individual covers, a vector $\mathbf{d} \in \mathbb{N}^{|T|}$ is constructed by concatenating target demands, and a matrix $\mathbf{A} \in \{0,1\}^{|T| \times |C|}$:

$$
a_{t,c} = \begin{cases} 1, & \text{if } c \text{ covers } t \\ 0, & \text{otherwise}\,. \end{cases}
\tag{4.22}
$$

It is evident, that the ordering of covers in $\mathbf{x}$ influences the values of $\mathbf{A}$. When the covers are ordered in a way that their right endpoints are non-decreasing, the columns of $\mathbf{A}$ consist of consecutive ones independent of the target order. This is already proved in Theorem 4.5. Because the matrix $\mathbf{A}$ has the consecutive ones property it is TU. This fact combined with right-hand side vector $\mathbf{d}$ consisting of integers shows that the LP (4.21) has total dual integrality property and thus it has integral optimum. □

Theorem 4.13. Problems with single-interval covers, and single-interval targets without containment with positive integer-valued target demands, which can be formulated as (1-$\mathbb{I}$ | 1-$\mathbb{I}$; no containment, demand $\in \mathbb{N}$), have integral optimum.

Proof. The proof is practically identical to the proof of Theorem 4.12 and thus we provide only a quick outline. The difference is that the targets are ordered instead of covers, resulting in rows with consecutive-ones property according to Theorem 4.4. Consequently, the matrix $\mathbf{A}$ is TU and has an integral optimum. □

4.3.4 Target Separability

The following constraint changes the way how the containment relationship is defined. The target separability represents the concept that an arbitrary target t is contained by a cover c when at least one of the target's single intervals $t' \in t$ is a subset of c:

$$\exists t' \in t : t' \subseteq c. \tag{4.23}$$

The keyword "separable" located among the target constraints represents the target separability constraint:

$$(\mathbb{C} \mid \mathbb{T}; \text{separable}). \tag{4.24}$$

Upon further inspection, it becomes evident that the introduction of target separability does not have any impact on problems that have only single interval targets 1-$\mathbb{I}$. Therefore, we can write that the family (1-$\mathbb{I}$; cost | 1-$\mathbb{I}$; separable) is solvable in polynomial time. When the target separability constraint is combined with the target demand the problem formulation (4.20) does not change, and ($\mathbb{I}$ | $\mathbb{I}$; demand, separable) is solvable in polynomial time using LP. Once again because the addition of target separability constraint does not impact the problems with single interval targets, we conclude that the families (1-$\mathbb{I}$; no containment | 1-$\mathbb{I}$; demand $\in \mathbb{N}$, separable) and (1-$\mathbb{I}$ | 1-$\mathbb{I}$; no containment, demand $\in \mathbb{N}$, separable) have integral optimum.

4.4 Computing Block Measurement Frequencies

Now that we have successfully created blocks we must know how many times each block has to be inserted into a tuning plan for the complete realization of every track and survey request. To reformulate, we need to find the goal measurement frequency of each block such that all track requests and sub-surveys are measured often enough while minimizing the receiver usage. This problem can be transformed into the MICntP described by a family ($\mathbb{I}$; cost, type | $\mathbb{I}$; demand, separable, size). The cover cost, target demand, and target separability constraints were already introduced in the preceding chapter. In this chapter, we have decided to describe the two remaining constraints - the type and cover size - since they are closely tied to our problem, relatively simple, and not general enough to be included in the previous one.

The *type* constraint limits which targets can be contained by which covers based on their types. Each cover c is assigned type $\mathit{type}(c)$ from the set of all possible cover types $\mathcal{T}_{\text{cover}}$. A similar is done for each target t, where $\mathit{type}(t) \in \mathcal{T}_{\text{target}}$. Finally, a boolean function $\mathit{typeMeasurable} : \mathcal{T}_{\text{cover}} \times \mathcal{T}_{\text{target}} \to \{0, 1\}$ is defined and determines whether the cover type could contain the stated target type. The type constraint is in the MICntP notation denoted by the keyword "type":

$$(\mathbb{C}; \text{type} \mid \mathbb{T}). \tag{4.25}$$

The last remaining constraint - the *cover size* - also changes the definition of containment relationship. This is accomplished by setting an upper limit on the total size of covers that contain the target. For every single interval of every target $t' \in t$ the problem formulation is extended by its cover size limit $\mathit{size}(t') \in \mathbb{R}_{\geq 0}$ such that $\mathit{size}(t') \geq |t'|$. The "size" keyword describes the presence of size constraint in the MICntP.

$$(\mathbb{C} \mid \mathbb{T}; \text{size}) \tag{4.26}$$

Finally, when we combine all of these constraints, the cover c, composed of single intervals $c' \in c$, contains the target t or t is contained by c, only if the cover's type can measure the target's type, at least one single interval belonging to the target $t' \in t$ is contained by the cover and the sum of sizes of cover's single intervals that intersect with t' is lesser or equal to $\mathit{size}(t')$. This is described by the following equation:

$$\mathit{typeMeasurable}(\mathit{type}(c), \mathit{type}(t)) \wedge \left(\exists t' \in t : \Big(t' \subseteq c \Big) \wedge \left(\sum_{\substack{c' \in c: \\ c' \cap t' \neq \emptyset}} |c'| \leq \mathit{size}(t') \right) \right). \tag{4.27}$$

Fortunately, the problem family ($\mathbb{I}$; cost, type | $\mathbb{I}$; demand, separable, size) remains solvable by LP in polynomial time because the only differences in problem formulation (4.20) caused by the redefinition of containment relationship are over which covers the summations are made, which does not influence its complexity.

The transformation of our problem to the MICntP is quite straightforward. First, the constructed blocks $\mathcal{B}$ are filtered to keep only those that realize a unique set of requests. This is done to decrease the constraint matrix's size and the number of optimal solutions, which should reduce the LP solver runtime. Then, for each of the unique blocks $B \in \mathcal{B}$, we create a multiple-interval cover c equal to the block's configuration $C(B)$ with cover type $\mathit{type}(c)$ equal to the $\mathit{SNN}(B)$ and the cost defined as follows:

$$\mathit{cost}(c) = \begin{cases} 4, & \text{if } \mathit{type}(c) = \mathit{all} \\ 1, & \text{otherwise,} \end{cases} \tag{4.28}$$

where the value corresponds to the number of receivers occupied by the block in the tuning plan. Each track request $r_{\text{track}} \in \mathcal{R}_{\text{track}}$ corresponds to a multiple-interval target t whose single intervals $t' \in t$ are equal to the track's emitter frequencies $f(e)$ with cover size limits $\mathit{size}(t)$ set to $b_{\max}(e)$, its demand $\mathit{demand}(t)$ equals the goal measurement frequency $\widehat{m}(r_{\text{track}})$, and target type $\mathit{type}(t)$ is track. For each sub-survey sr a single interval target t is created such that it corresponds to the single interval $f(sr)$, the cover size limit $\mathit{size}(t)$ is set to infinity, the demand $\mathit{demand}(d)$ once again equals the goal measurement frequency $\widehat{m}(sr)$, and its target type $\mathit{type}(t)$ is survey.

The last thing remaining is to define a *typeMeasure* function. The block with the *SNN* set to *all* can realize both track requests and sub-surveys, in contrast, the "*any*" block can realize only sub-surveys. Therefore, we define the function as follows:

$$typeMeasurable(type(c), type(t)) = \begin{cases} 1, & \text{if } type(c) = all \\ 1, & \text{if } (type(c) = any) \wedge (type(t) = survey) \\ 0, & \text{otherwise.} \end{cases} \tag{4.29}$$

After the problem is transformed into the MICntP, it can be solved as LP using any available LP solver like those already mentioned above. Each of the decision variables x_c corresponds to a block B according to which it was created and describes its optimal goal measurement frequency $\widehat{m}(B)$.

4.5 Construction of Tuning Plan

The following step is to finally construct the tuning plan. The construction process can be formulated as an optimization problem described by Equation (4.30). The goal is to produce a tuning plan L that minimizes the sum of squares of the difference between the goal $\widehat{m}(B)$ and the realized $m(L, B)$ measurement frequency of each block $B \in \mathcal{B}$ such that each user request is scheduled, no noisy frequencies are observed, and frequencies are measured aperiodically.

$$\begin{aligned} \min_{L} \quad & \sum_{B \in \mathcal{B}} \max\{0, \widehat{m}(B) - m(L, B)\}^2 \\ \text{s.t.:} \quad & \text{All user requests are scheduled.} \\ & \text{Noise frequencies are never observed.} \\ & \text{Frequencies are measured aperiodically.} \end{aligned} \tag{4.30}$$

How we solve this problem is described by Algorithm 4.4. First, the empty tuning plan L of predetermined size l is initialized and each user request $r_{\text{user}} \in \mathcal{R}_{\text{user}}$ is directly inserted into it, as can be seen at Line 3. This ensures that all user requests are scheduled and consequently, the first constraint of (4.30) is fulfilled. Now, we can freely insert the constructed blocks because the second constraint is always satisfied due to the fact that no block contains noisy frequencies as described in Chapter 4.2. Since there are only two simple ways the block can occupy the tuning plan - one or four receivers - and it does not matter at which measurement interval we insert the block, we conclude that the blocks can be inserted into the tuning plan greedily. The greedy approach also ensures that the tuning plan construction is fast which is one of the main requirements for the planning algorithm.

During each step, the block with the highest priority is selected using the priority queue, as described at Line 9. The priority of block $B \in \mathcal{B}$ is formulated in Equation (4.31) and describes how much the criterion will decrease due to its insertion.

$$\begin{aligned} \mathit{priority}(B) &= \max\{0, \widehat{m}(B) - m(L,B)\}^2 - \max\left\{0, \widehat{m}(B) - m(L,B) - \frac{1}{l}\right\}^2 \\ &= (\widehat{m}(B) - m(L,B))^2 - \max\left\{0, \widehat{m}(B) - m(L,B) - \frac{1}{l}\right\}^2 \\ &= \begin{cases} (\widehat{m}(B) - m(L,B))^2, & \text{if } \left(\widehat{m}(B) - m(L,B) - \frac{1}{l}\right) \leq 0 \\ 2\widehat{m}(B)\frac{1}{l} - 2m(L,B)\frac{1}{l} - \frac{1}{l^2}, & \text{otherwise} \end{cases} \\ &= \begin{cases} (\widehat{m}(B) - m(L,B))^2, & \text{if } \left(\widehat{m}(B) - m(L,B) - \frac{1}{l}\right) \leq 0 \\ \frac{2}{l}(\widehat{m}(B) - m(L,B)) - \frac{1}{l^2}, & \text{otherwise} \end{cases} \end{aligned} \tag{4.31}$$

We assume that $\widehat{m}(B) - m(L,B)$ is larger than 0 because otherwise, the block would not be inserted into the queue. The left expression denotes the amount currently added by the block B to the criterion while the right one corresponds to the amount added after the next insertion of B. The only difference between these two expressions is $\frac{1}{l}$ which is equal to the amount of measurement frequency by which $m(L,B)$ increases after another insertion of B into the plan.

The algorithm then thrice attempts to insert the selected block B into the plan, as shown at Lines 10-24, and with each attempt the insertion conditions are less constraining. This is achieved by generating all possible positions into which the block can be inserted, these can vary only by the measurement interval and sensor nodes because the block already determines the receiver type and receivers of the same type are identical. During the first attempt, described at Lines 10-16, each position $P \in \mathcal{P}_1$ must fulfill the following conditions: empty, feasible, not present, no fragmentation, and no repetition.

The *empty* and *feasible* conditions are quite straightforward, no block can be scheduled there and it must be possible to insert the selected block into it. The block's *SNN* determines the meaning of the *not present* condition which aims to prevent the useless block repetition. When the need equals *all*, the position cannot be at the same measurement interval as the selected block was already scheduled during the previous iterations. Similarly, when the need is *any*, the generated position cannot be at the same measurement interval and sensor node as the selected block was previously inserted. The *no fragmentation* condition helps to create the compact tuning plan by filtering out positions at which the insertion of block B results in a decrease of the fragmentation count. This count determines how many blocks with *SNN* set to *all* can be inserted into the tuning plan L at the measurement interval τ and it is possible to denote as follows:

$$\mathit{fragmentCount}(L,\tau) = \sum_{RT \in \mathbf{RT}} \min_{i \in \{1,\ldots,|\mathbf{SN}|\}} \sum_{R \in \mathbf{R}_i} [\![RT = RT(R)]\!] [\![L_{\tau,R} \text{ is empty}]\!]. \tag{4.32}$$

Consequently, the blocks with *SNN* equal to *any* tend to cluster into the same measurement intervals while spreading to different sensor nodes resulting in a compact plan with more positions for future insertions of *"all"* blocks. Finally, the *no repetition* condition depends on the set of previously constructed tuning plans $\mathcal{H}$ which is one of the algorithm's inputs. The condition allows only for positions at the measurement intervals during which the selected block was not scheduled in previously constructed tuning plans from $\mathcal{H}$. This helps to mitigate the periodicity between the current and historical plans.

The second attempt, which is described at Lines 17-20, constructs position set $\mathcal{P}_2$ with the same conditions as the first, except for *no repetition*. The final attempt, shown at Lines 21-24, removes one more condition - *no fragmentation* - and constructs the set of positions $\mathcal{P}_3$. When any of the position sets $\mathcal{P}_1$, $\mathcal{P}_2$, or $\mathcal{P}_3$ is not empty, one of the positions P is randomly sampled, and the block is inserted into the tuning plan L at this position. Random sampling practically ensures the aperiodicity of the constructed plan and thus we consider the last constraint of problem (4.30) satisfied. Then the block's priority is updated and if it is larger than zero, the block is returned to the priority queue with this new priority. These block insertions are repeated until the criterion is zero or no further blocks can be scheduled.

If the algorithm's main loop has finished and the tuning plan is not full, the randomly sampled blocks fill the remaining empty positions as described in Algorithm 4.6. The following steps are repeated until the set of blocks $\mathcal{B}$ is empty. A block B is randomly sampled from the blocks set $\mathcal{B}$. Its set of all possible positions $\mathcal{P}$ is generated according to the constraints: empty, feasible, and no repetition. If this set is empty, the block B is removed from set $\mathcal{B}$, otherwise, the block is inserted at the position P sampled from $\mathcal{P}$.

4.6 Update of Sub-surveys

As already mentioned, we expect that the tuning plan will be constructed repeatedly and the request exists many times longer than the duration of a single plan. Therefore, it seems reasonable to assume that sometimes the track requests or sub-surveys might not be realized often enough, when the system becomes overwhelmed with requests, and, on the other hand, during idle periods they might be measured more often than necessary. It may be useful to include information about the previous measurements in the goal measurement frequencies of requests to help balance these extremes. Fortunately, this is already implemented for track requests because variations in realized measurement frequency directly influence the information gain of a target and consequently the computation of its goal measurement frequency during the next run of the algorithm. This chapter describes how the algorithm approaches this problem for sub-surveys.

Since the recent deviations are significantly more important than the old ones, we have decided to collect the discounted previous measurement frequencies m^{hist} with the discount

Algorithm 4.4: Tuning plan construction

```
Input: size of tuning plan $l$, set of blocks $\mathcal{B}$, set of user requests $\mathcal{R}_{\text{user}}$, set of
       previous tuning plans $\mathcal{H}$
Output: tuning plan $L$
Initialize empty tuning plan $L$ with size $l$
forall $r_{\text{user}} \in \mathcal{R}_{\text{user}}$ do
    Insert $r_{\text{user}}$ into $L$ at predetermined slots
Initialize empty priority queue $Q$
forall $B \in \mathcal{B}$ do
    if $\mathit{priority}(B) > 0$ then
        Insert block $B$ to $Q$ with $\mathit{priority}(B)$
while $Q$ is not empty do
    $B \leftarrow$ Pop block with highest priority from $Q$
    $\mathcal{P}_1 \leftarrow \{$
        $P \in L \mid$
        for $B$ : empty; feasible; not present; no fragmentation; no repetition
    $\}$
    if $\mathcal{P}_1$ is not empty then
        $L, Q \leftarrow$ InsertBlock($L$, $Q$, $B$, $\mathcal{P}_1$)
        continue
    $\mathcal{P}_2 \leftarrow \{P \in L \mid$ for $B$ : empty; feasible; not present; no fragmentation$\}$
    if $\mathcal{P}_2$ is not empty then
        $L, Q \leftarrow$ InsertBlock($L$, $Q$, $B$, $\mathcal{P}_2$)
        continue
    $\mathcal{P}_3 \leftarrow \{P \in L \mid$ for $B$ : empty; feasible; not present$\}$
    if $\mathcal{P}_3$ is not empty then
        $L, Q \leftarrow$ InsertBlock($L$, $Q$, $B$, $\mathcal{P}_3$)
        continue
if $L$ is not full then
    $L \leftarrow$ FillEmptySlots($L$, $\mathcal{B}$)
```

Algorithm 4.5: InsertBlock

```
Input: tuning plan $L$, priority queue $Q$, block $B$, set of positions $\mathcal{P}$,
Output: tuning plan $L$, priority queue $Q$
1 $P \leftarrow$ Randomly sample $\mathcal{P}$
2 Insert block $B$ to $L$ at position $P$
3 Update $priority(B)$
4 if $priority(B) > 0$ then
5     Insert block $B$ to $Q$ with $priority(B)$
```

Algorithm 4.6: FillEmptySlots

```
Input: tuning plan $L$, set of blocks $\mathcal{B}$,
Output: tuning plan $L$
1 while $\mathcal{B}$ is not empty do
2     $B \leftarrow$ Randomly sample $\mathcal{B}$
3     $\mathcal{P} \leftarrow \{P \in L \mid$ for $B$ : empty; feasible; not present$\}$;
4     if $\mathcal{P}$ is not empty then
5         $P \leftarrow$ Randomly sample $\mathcal{P}$
6         Insert block $B$ to $L$ at position $P$
7     else
8         Remove $B$ from $\mathcal{B}$
```

factor $\gamma \in (0;1)$. Let us consider the situation when the algorithm has constructed the h-th tuning plan, its historical measurement frequencies m_h^{hist} look as follows:

$$m_h^{\text{hist}} = \gamma^{h-1} m_1 + \cdots + \gamma m_{h-1} + m_h = \sum_{i=1}^{h} \gamma^{i-1} m_i, \tag{4.33}$$

where m_i symbolizes the measurement frequency realized by i-th plan. We consider that the highest possible measurement frequency is 1 and consequently, it holds that:

$$m_h^{\text{hist}} \leq \frac{1-\gamma^h}{1-\gamma}. \tag{4.34}$$

The algorithm's goal for the $(h+1)$-th iteration is to realize the corresponding sub-survey request often enough so that the normalized historical measurement frequency after that run is larger or equal to the overall goal measurement frequency $\widehat{m}$ determined by its survey request. Therefore, the measurement frequency during the $(h+1)$-th iteration

m_{h+1} must be:

$$\begin{aligned}
\widehat{m} &\leq \frac{\gamma^h m_1 + \cdots + \gamma m_h + m_{h+1}}{\frac{1-\gamma^{h+1}}{1-\gamma}} \\
\frac{1-\gamma^{h+1}}{1-\gamma}\widehat{m} &\leq \gamma^h m_1 + \cdots + \gamma m_h + m_{h+1} \\
m_{h+1} &\geq \frac{1-\gamma^{h+1}}{1-\gamma}\widehat{m} - \gamma(\gamma^{h-1} m_1 + \cdots + m_h) \\
m_{h+1} &\geq \frac{1-\gamma^{h+1}}{1-\gamma}\widehat{m} - \gamma m_h^{\text{hist}}.
\end{aligned} \tag{4.35}$$

And the goal measurement frequency of the sub-survey during the construction of $(h+1)$-th tuning plan should be set to $\frac{1-\gamma^{h+1}}{1-\gamma}\widehat{m} - \gamma m_h^{\text{hist}}$.

The above-stated process is done for each sub-survey $sr \in \mathcal{S}(r_{\text{survey}})$ of every survey request $r_{\text{survey}} \in \mathcal{R}_{\text{survey}}$ to determine its goal measurement frequency for next iteration $\widehat{m}(sr)$ as can be seen in Algorithm 4.7. First, the historical measurement frequency $m^{\text{hist}}(sr)$ is updated according to the number of its realizations in a recently constructed tuning plan $m(L, sr)$. Then the goal measurement frequency is computed as described in Equation (4.35) and rounded to lie between 0 and 1. This ensures that the previous sub-survey measurement frequencies are reflected during the future construction of the tuning plan.

Algorithm 4.7: Sub-surveys update

```
Input: survey requests $\mathcal{R}_{\text{survey}}$, tuning plan $L$
Output: goal measurement frequency $\widehat{m}(sr)\ \forall sr \in \mathcal{S}(r)\ \forall r \in \mathcal{R}_{\text{survey}}$
1 forall $r_{\text{survey}} \in \mathcal{R}_{\text{survey}}$ do
2     forall $sr \in \mathcal{S}(r_{\text{survey}})$ do
3         $m^{\text{hist}}(sr) \leftarrow \gamma m^{\text{hist}}(sr) + m(L, sr)$
4         $\widehat{m} \leftarrow \frac{1-\gamma^{h+1}}{1-\gamma}\widehat{m}(r_{\text{survey}}) - \gamma m^{\text{hist}}(sr)$
5         $\widehat{m}(sr) \leftarrow \max\{0, \min\{\widehat{m}, 1\}\}$
```

Chapter 5

Experimental Results

The complete evaluation of the algorithm proposed in the previous chapter would require us to develop complex scenarios and a simulation environment where the constructed tuning plans would be executed according to these scenarios. As already mentioned, this would be highly time-consuming and far beyond the scope of this thesis. Besides, thanks to the careful algorithm design, we already know that the produced tuning plans achieve all given constraints and realize as many requests as possible. Therefore, we have decided to limit experiments conducted in this chapter to simpler ones focused on the algorithm's other essential property - computation time and its associated metrics. Each of these simpler experiments corresponds to a single run of the algorithm which produces one tuning plan. Furthermore, the track measurement frequencies are assigned during the request creation and consequently, the algorithm's first step - computation of track measurement frequencies - is skipped. The experiments are designed so we can study the effect of different requests on the execution time with a focus on the processing time of the LP solver since it is the expected bottleneck of our algorithm.

We must point out that adding user requests does not influence the complexity of the solved LP and its influence on the total computation time is insignificant. A similar can be said about noise requests because the only step of the algorithm influenced by noise is the block construction which is heuristic and minimally affects the computational complexity. Therefore, we do not include these two request types in any of the following experiments. In contrast, the track and survey request types directly influence the amount and character of generated blocks and thus also the number of targets, covers, and LP complexity.

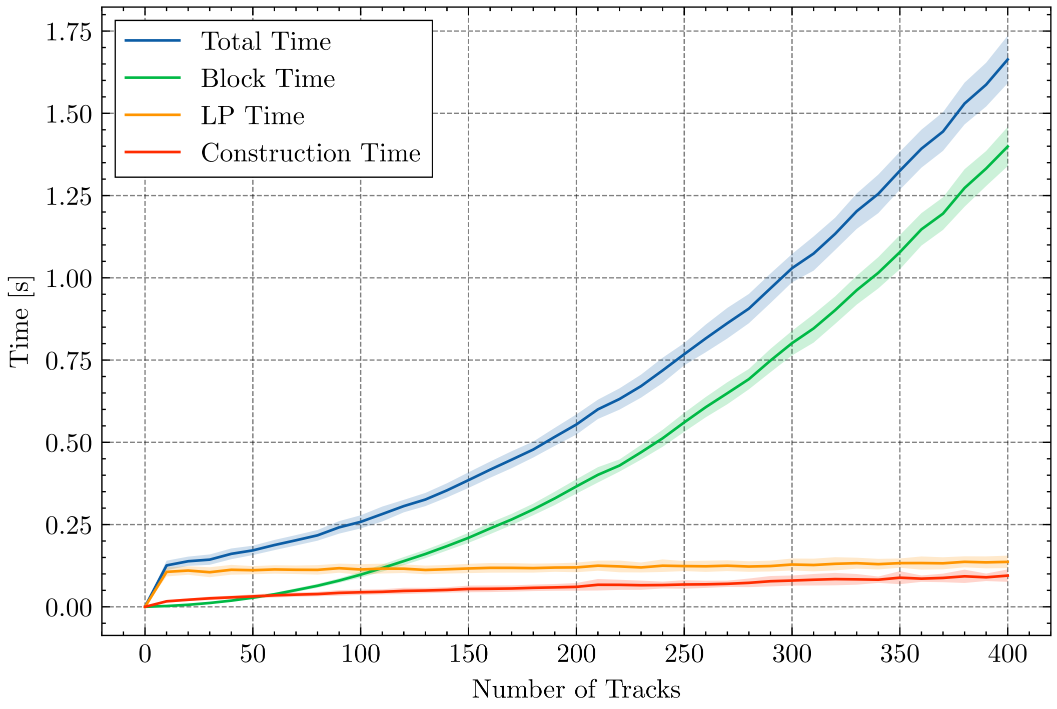
Because no publicly available database of track requests and their emitter frequencies exists, we have decided that all of the track requests used in the experiments will be randomly generated. During each experiment, all track requests $\mathcal{R}_{\text{track}}$ have the same predetermined number of emitters. The single intervals corresponding to emitters are uniformly distributed across all observable frequencies. The size of each of these single intervals is also sampled from a uniform distribution with possible values ranging from 1MHz to the maximal possible value such that there exists at least one configuration that can realize the corresponding emitter. The maximum configuration size $b_{\max}(e)$ of emitter e is with the probability $P(b_{\max})$ equal to the sum of the emitter's size and a value sampled from the geometric distribution with success probability parameter equal to 0.02, otherwise it is set to infinity. Finally, the goal measurement frequencies are sampled from a uniform distribution $\mathcal{U}(0, 1)$ and then normalized such that $\sum_{r \in \mathcal{R}_{\text{track}}} \widehat{m}(r) = 1$. Consequently, the system should always have enough resources to realize all generated track requests.

Every one of the following subchapters proposes a set of experiments that focus on a different aspect of track or survey request to study its influence on the algorithm's runtime. All presented experiments were repeated 100 times with different input values to obtain reliable results. Hence, the following figures display the sampled mean values while the shaded regions correspond to the standard deviation, as is usual. During the experiments, we measure the execution time of the block generation combined with the computation of containment relationship between covers and targets (block time), the execution time of LP solver (LP time), and the execution time of tuning plan construction (construction time). The total execution time (total time) is also measured which is approximately equal to the sum of the aforementioned run times. Moreover, we keep track of the number of blocks that realize unique combinations of requests (unique block count), which influences the size of the created LP. The number of blocks with resulting non-zero goal measurement frequency (non-zero block count) is also monitored to determine how many unique blocks will be inserted into the tuning plan.

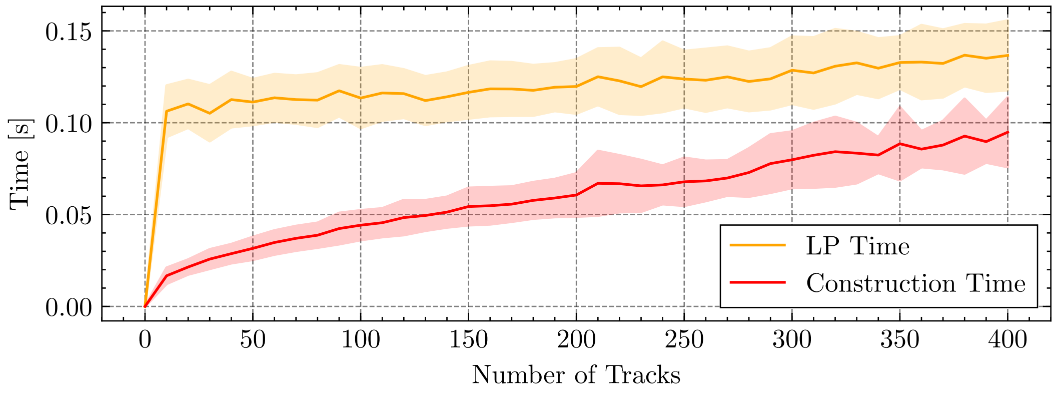
The algorithm was implemented in Python 3.12. The portion library [37] was used for the interval representation and various operations over intervals that often appear in the algorithm. The LP used to compute the goal measurement frequencies of blocks as described in Chapter 4.4 was formulated with the help of Pyomo package [38][39] and subsequently solved by the COIN-OR Branch-and-Cut (CBC) solver [40]. The NumPy [41] was used to compute the statistics over data collected during the experiments. These statistics were then plotted using the Matplotlib library [42] according to the figure format provided by the SciencePlots package [43]. All of the experiments were conducted on the laptop with 13th Gen Intel(R) Core(TM) i9-13980HX 2.20GHz CPU, NVIDIA(R) GeForce RTX(TM) 4070 Laptop GPU 8GB GDDR6, and 32GB of DDR5-5600 RAM.

5.1 Number of Track Requests

The first set of experiments focuses on how the number of track requests influences the algorithm runtime. The experiments are set up so that the number of track requests is an independent variable that increases from 0 to 400 with steps of size 10. Each track request has only one emitter and $P(b_{\max})$ equals 0.0. During these experiments, no survey requests are inserted into the algorithm.



(a)



(b)

Figure 5.1: Measured execution times as a function of the number of track requests. Figure (b) displays a detail of (a) for better legibility.

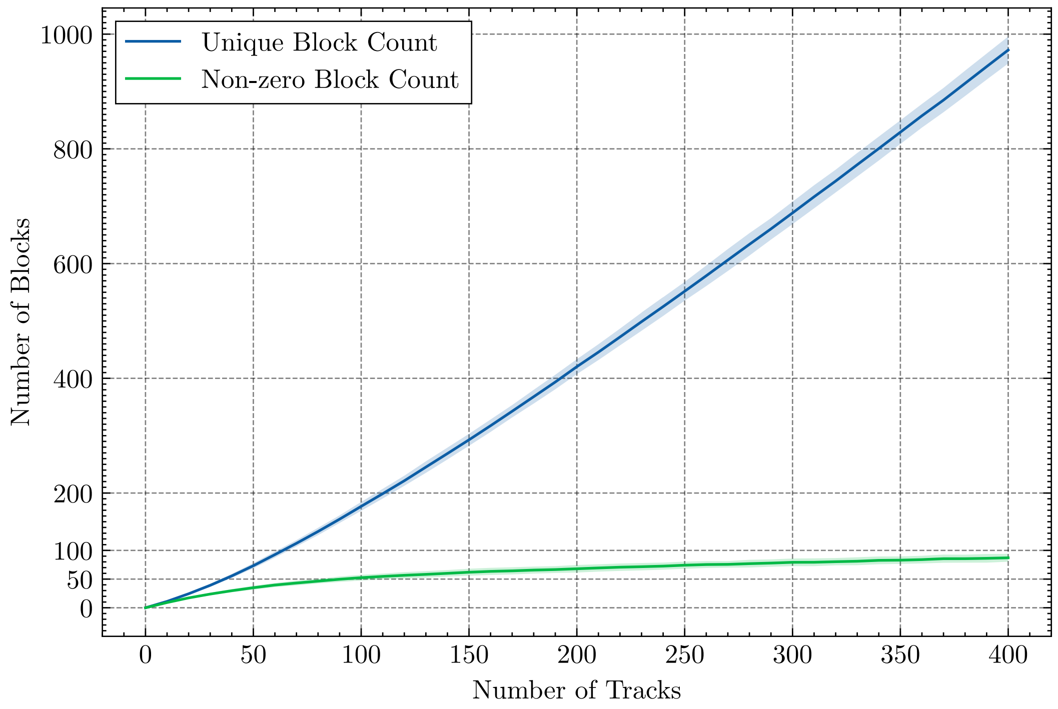


Figure 5.2: Unique and non-zero block counts as a function of the number of track requests.

The measured execution times can be seen in Figure 5.1. Surprisingly, the LP time (orange) seems to grow linearly with the number of track requests. The slope of this growth is very gentle, starting at around 0.105s and ending at 0.135s while the number of tracks increases from 10 to 400. We speculate that this favorable outcome is caused by the interconnected character of generated blocks which later form the constraint matrix of the solved LP. The construction time (red) is throughout the experiments shorter than the LP time but its growth is steeper. During the majority of the experiments, the most time-consuming part of the experiment was the block construction and containment computation (green) whose execution time grew more than linearly. This was not caused by the amount of generated blocks but by the computation of the containment relationship between covers and all existing targets. Fortunately, it can be mitigated in a real-world deployment by memorizing the constructed covers with targets they contain and reusing them in future algorithm runs. This is possible because we expect that requests are added gradually throughout the multiple runs of the algorithm while each request generates the same blocks and therefore the same covers.

As mentioned in Chapter 4.2, the number of blocks generated per each track request is linear, and thus the number of both the unique and non-zero blocks has a linear upper bound. The amounts of blocks encountered during the experiments are plotted in Figure 5.2. It can be seen that the incline of the unique block count (blue) increases together with the number of tracks. This can be expected because when the algorithm has only a few randomly generated track requests that are sparsely distributed, the created blocks likely realize only their parent requests. And as the number of tracks increases the more unique request combinations arise. In contrast, the number of non-zero blocks behaves almost logarithmically which is promising in terms of the tuning plan construction.

5.2 Number of Emitters per Track Request

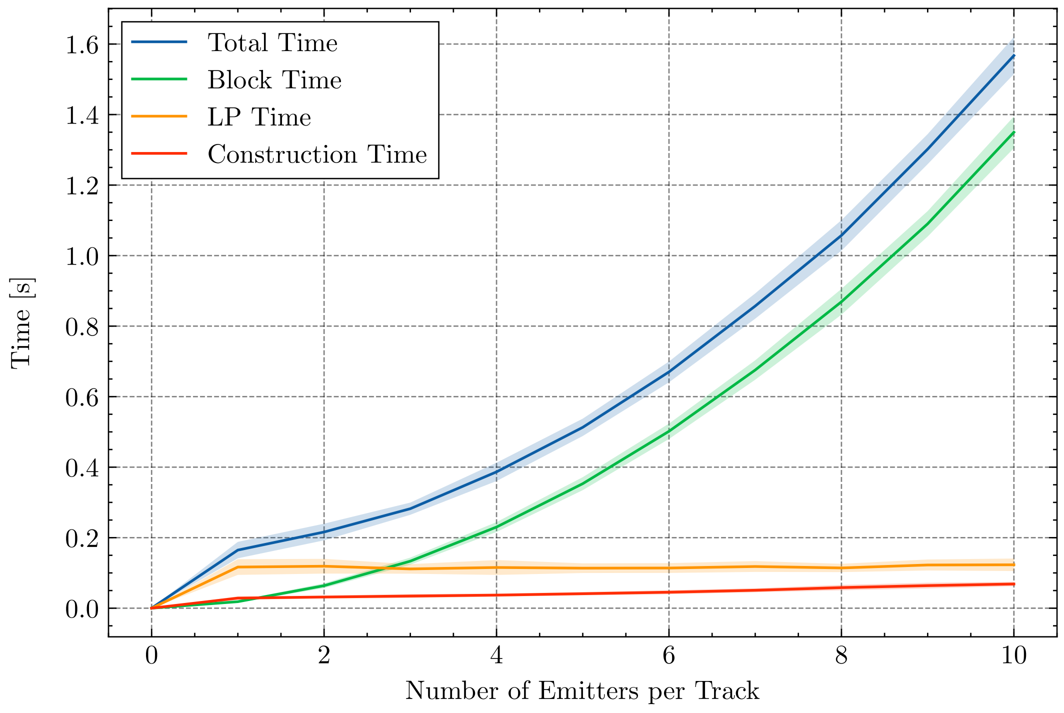
As the name suggests, this subchapter examines the dependence between the algorithm's performance and the number of emitters per track request. The number of generated track requests is the same in all experiments and equals 40. The number of emitters assigned to each track request is an explanatory variable that ranges from 0 to 10. Similar to the previous chapter, $P(b_{\max})$ is set to 0.0 and no survey requests are inserted into the algorithm.

Figure 5.3 displays the execution times observed during the experiments. The LP time is practically constant which is likely caused by the fact that the number of track requests is not increasing, therefore the size of constructed LP remains the same across all of the experiments. The construction time increases linearly with the addition of further emitters to the tracks. We already know that the number of generated blocks directly depends on the overall amount of emitters in the experiment. Because of this and the fact that the current and previous scenarios have equal emitter counts, the block times of both scenarios behave the same. The number of unique blocks also behaves similarly to the previous subchapter, as shown in Figure 5.4. On the other hand, the number of non-zero blocks slowly decreases. This is because, with each additional emitter per track request, the chance that one block can realize multiple requests increases, and consequently the algorithm can fulfill all of the requests with fewer blocks.

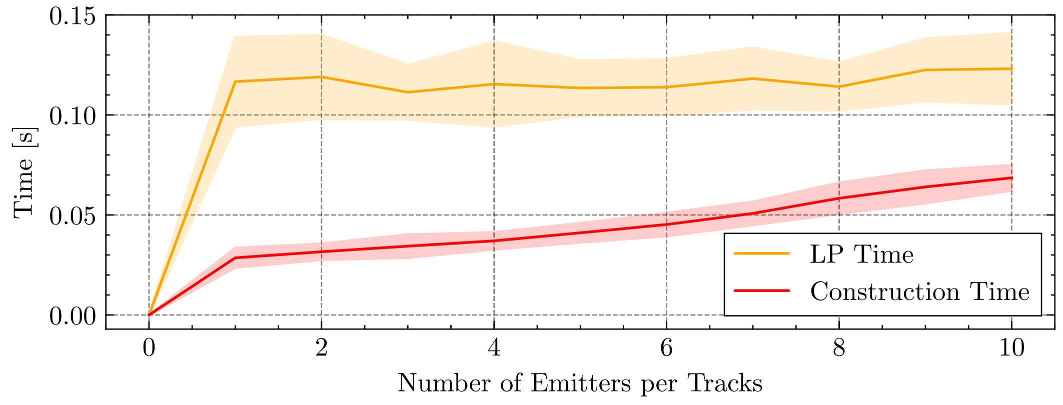
5.3 Probability of Emitter Configuration Size

The last property of track request whose influence on the algorithm remains to be explored is the probability of emitter configuration size, which was already denoted as $P(b_{\max})$. We consider eleven different probability values that range from 0.0 to 1.0 with steps of size 0.1. The number of track requests during this set of experiments is constant and equals 200. Each track has only one emitter. No other requests are inserted into the algorithm except the already mentioned track requests.

As can be seen in Figure 5.5, both the LP and construction time remain constant during all experiments. Therefore, we conclude that introducing the configuration size limits for emitters does not influence the complexity of the constructed LP. Figure 5.6 shows that the unique block count decreases as $P(b_{\max})$ grows. This happens because the introduced size limits restrict the size of generated blocks which cannot realize as many unique requests

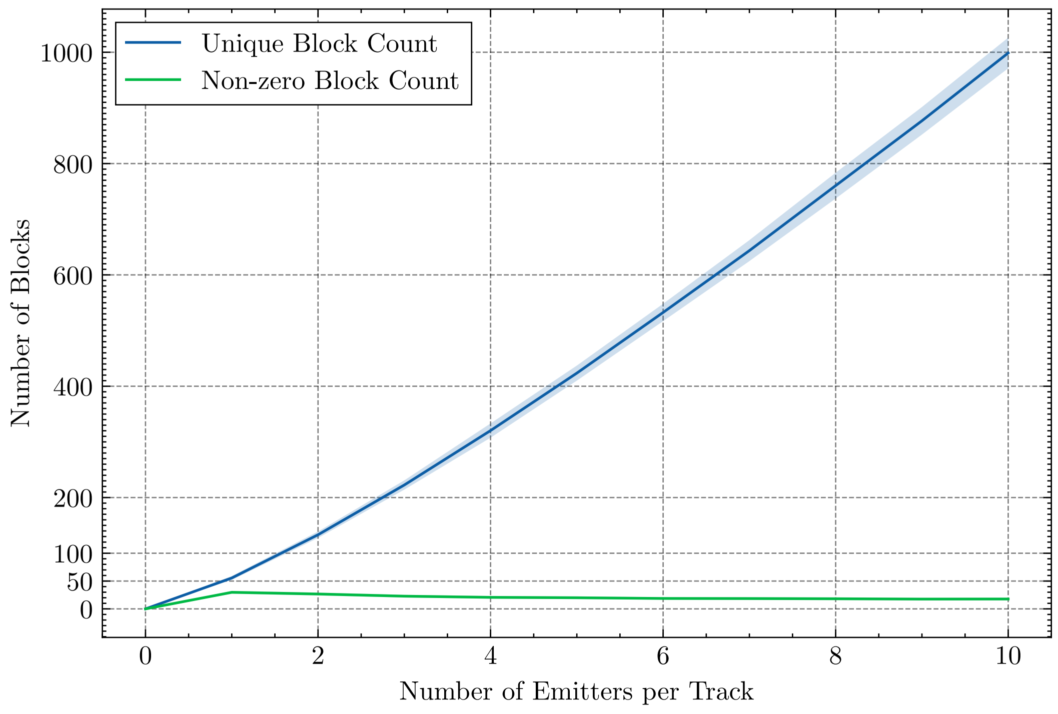


(a)

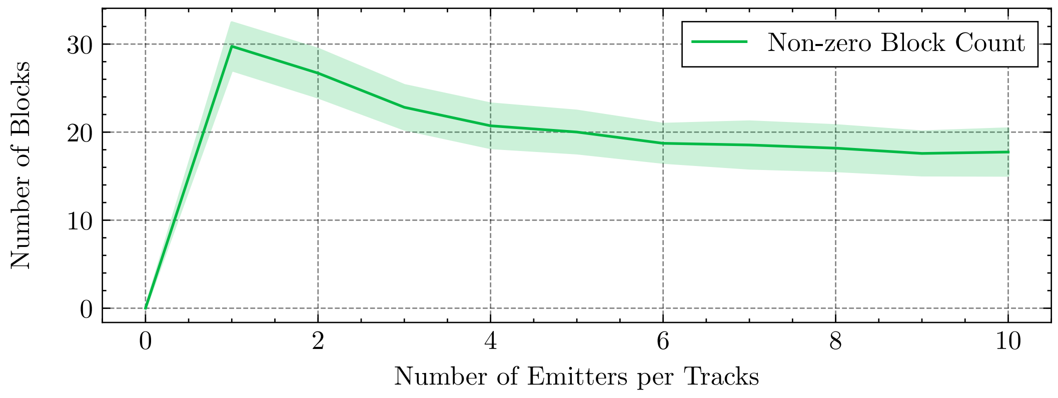


(b)

Figure 5.3: Measured execution times as a function of the number of emitters per track request. Figure (b) displays a detail of (a) for better legibility.



(a)



(b)

Figure 5.4: Unique and non-zero block counts as a function of the number of emitters per track request. Figure (b) displays a detail of (a) for better legibility.

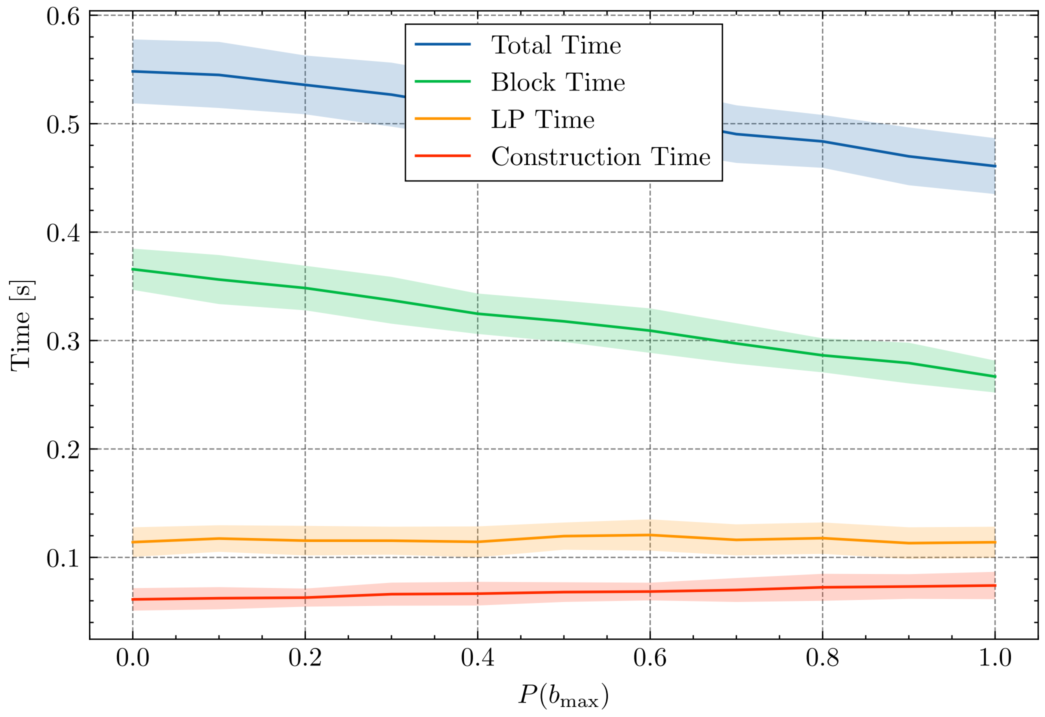


Figure 5.5: Measured execution times as a function of the probability of emitter configuration size.

as usual. Consequently, the block time shrinks with increasing $P(b_{\max})$. In contrast, the number of non-zero blocks must increase to realize all requests.

5.4 Complete Survey Request

In this chapter, we focus on how adding survey requests changes the algorithm's behavior. Since the monitored frequencies of survey requests cannot overlap and splitting a survey request into sub-surveys is the same for all requests, the change in the algorithm's complexity cannot be caused by the number of requests but by the total size of monitored frequencies. Therefore, we have decided to use only a single survey request that spans all measurable frequencies which we will compare with the results presented in Chapter 5.1. The goal measurement frequency of this request is set to 0.1. Other than that, the inputs of the experiments are identical to those in the first set of experiments. The track request count ranges between 0 and 400 with steps of 10, each of the track requests has one emitter, and $P(b_{\max})$ equals 0.0.

From Figure 5.7 we see that the block time significantly increased compared to the first scenario. This is expected because the number of generated blocks increases linearly with the newly added sub-surveys. In contrast with the experiments without the survey, the

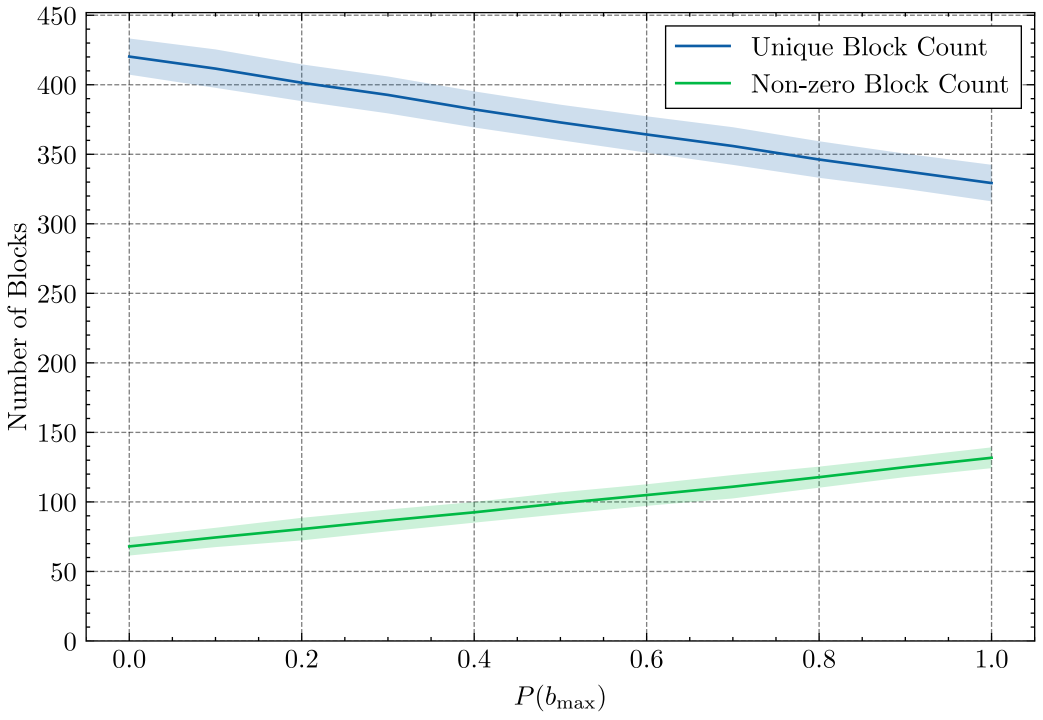
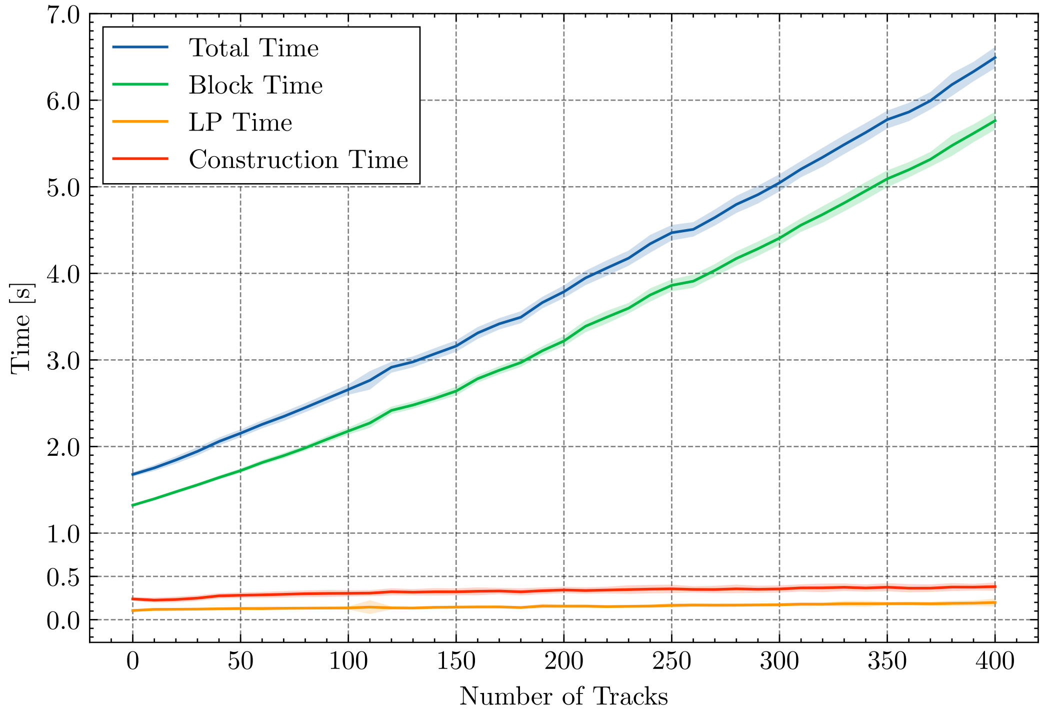
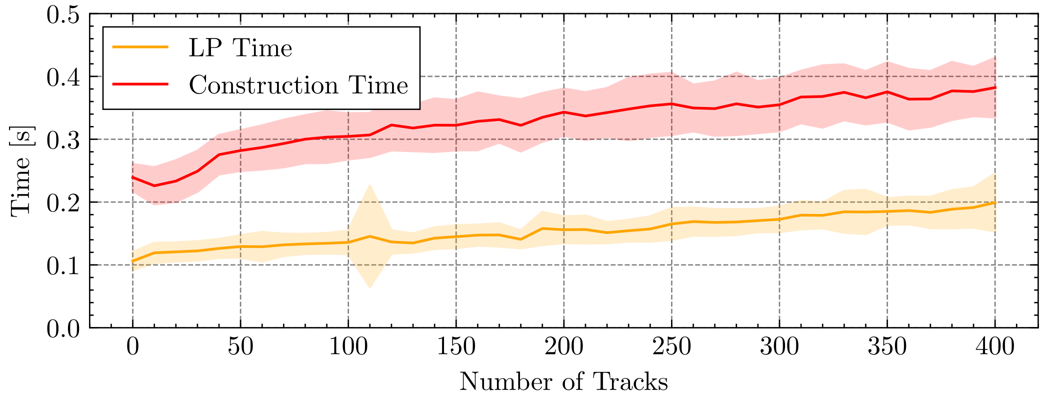


Figure 5.6: Unique and non-zero block counts as a function of the probability of emitter configuration size.

construction time is always longer than the LP time and grows almost logarithmically with the number of track requests. The LP time begins at around 0.1s and linearly increases to 0.2s as more track requests are added. The number of unique blocks also rises linearly, as plotted in Figure 5.8. The behavior of non-zero block count is logarithmic similar to the first scenario.



(a)



(b)

Figure 5.7: Measured execution times as a function of the number of track requests during a complete survey. Figure (b) displays a detail of (a) for better legibility.

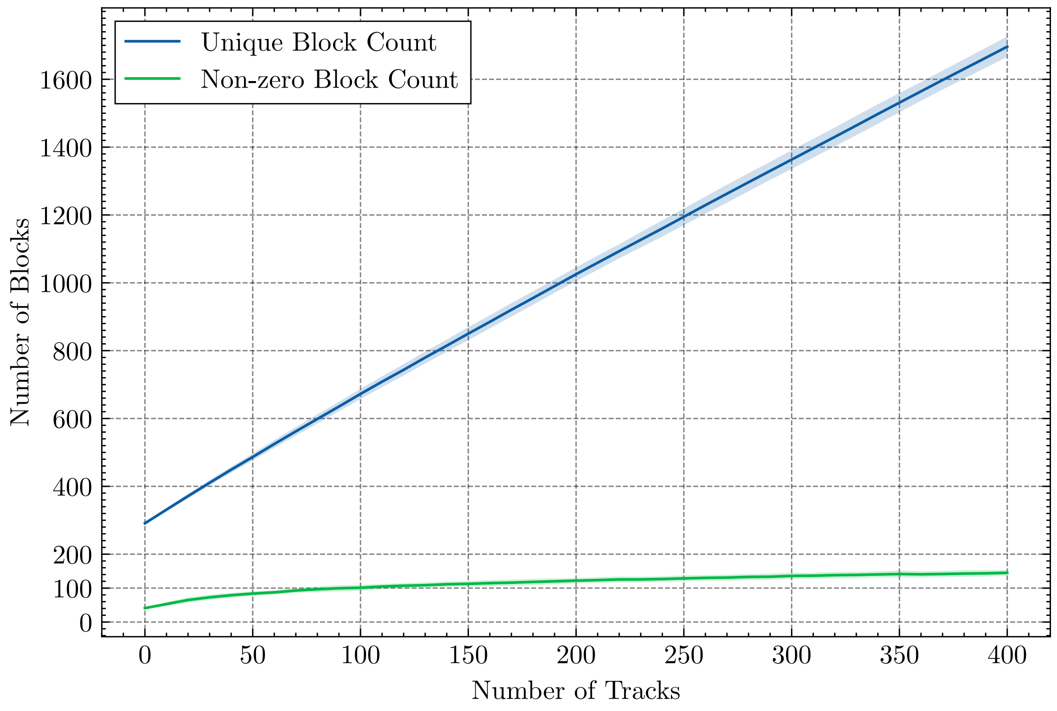


Figure 5.8: Unique and non-zero block counts as a function of the number of track requests during a complete survey.

Chapter 6

Conclusions

In this thesis, we proposed the algorithm for the construction of optimized tuning plans for a generalized passive radar system. Tuning plans produced by the algorithm are always feasible w.r.t. system's requests thanks to its careful design. One of the algorithm's steps optimizes receiver configurations to maximize the number of simultaneously realized requests which we consider to be our main contribution to the field of passive radar optimization. The receiver optimization is a special case of the newly formulated MICntP which can be optimally solved in polynomial time using LP. Furthermore, we studied other versions of MICntP with various constraints and proved that single-interval instances with cover costs can be solved in polynomial time. On the other hand, the problem becomes $\mathcal{NP}$-hard even when the targets remain single intervals and at least some of the covers are 2-intervals.

The experiments have focused on a relationship between the algorithm's runtime and request complexity during the construction of a single tuning plan. They have shown that the most time-consuming step of the algorithm is determining the coverage of targets by covers whose runtime grows quadratically with the number of requests. However, this can be mitigated in actual deployment by memorizing the information obtained during the construction of previous tuning plans. Surprisingly, the time it takes the optimizer to solve the constructed LP is short and increases linearly. A similar can be said about the step that constructs a tuning plan from generated blocks.

Future work should mostly focus on further experimentation. It would be ideal to integrate the proposed algorithm into the existing passive radar system and thoroughly test its behavior using a simulator capable of emulating real-world scenarios.

In the current algorithm, more specialized requests produced by the radar's subsystems are expected to be communicated via user requests. Instead of this process, it might be useful to add new request types which will simplify the communications from subsystems to the planner.

From the scheduling point of view, it might be interesting to study the optimality of the greedy algorithm used for inserting blocks into the tuning plan.

Bibliography

[1] Kulmon, P.; Suja, J.; Benko, M. Scheduling of Multi-Function Sensor. *IEEE Transactions on Radar Systems*, volume 1, 2023: pp. 729–739, doi:10.1109/TRS.2023.3335208.

[2] Skolnik, M. An introduction and overview of radar. *Radar Handbook*, volume 3, 2008: pp. 1–1.

[3] Richards, M. A.; Scheer, J.; Holm, W. A.; et al. Principles of modern radar. 2010.

[4] Rahman, H. *Fundamental Principles of Radar*. CRC Press, 2019.

[5] Guarnieri, M. The Early History of Radar [Historical]. *IEEE Industrial Electronics Magazine*, volume 4, no. 3, 2010: pp. 36–42, doi:10.1109/MIE.2010.937936.

[6] Goodchild. The 360ft transmitter towers at Bawdsey Chain Home radar station, Suffolk. May 1945, for Royal Air Force, image created and released by Imperial War Museum, [Online; accessed January 24, 2024]. Available from: `https://www.iwm.org.uk/collections/item/object/205196697`

[7] RADAR COVER, SEPTEMBER 1939 AND SEPTEMBER 1940. 1953, british Official Histories (History of the Second World War), [Online; accessed January 24, 2024]. Available from: `https://commons.wikimedia.org/wiki/File:Chain_home_coverage.jpg`

[8] Sarkar, T. K.; Salazar Palma, M.; Mokole, E. L. Echoing Across the Years: A History of Early Radar Evolution. *IEEE Microwave Magazine*, volume 17, no. 10, 2016: pp. 46–60, doi:10.1109/MMM.2016.2589200.

[9] Gortyna. Meteorological radar on the top of Praha hill in Brdy mountains. July 2016, [Online; accessed February 2, 2024]. Available from: `https://commons.wikimedia.org/wiki/File:Praha_radar4.jpg`

[10] ČHMÚ nowcasting webportal. February 2024, provided by Czech Hydrometeorological Institute, [Online; accessed February 2, 2024]. Available from: `https://www.chmi.cz/files/portal/docs/meteo/rad/inca-cz/short.html`

[11] Budge, M. C.; German, S. R. *Basic RADAR analysis*. Artech House, 2020.

[12] Griffiths, H. D.; Baker, C. J. *An introduction to passive radar*. Artech House, 2022.

[13] Kuschel, H.; Cristallini, D.; Olsen, K. E. Tutorial: Passive radar tutorial. *IEEE Aerospace and Electronic Systems Magazine*, volume 34, no. 2, 2019: pp. 2–19.

[14] Qu, Z.; Ding, Z.; Moo, P. A Radar Task Scheduling Method Using Random Shifted Start Time with the EST Algorithm. In *2019 IEEE Radar Conference (RadarConf)*, 2019, pp. 1–5, doi:10.1109/RADAR.2019.8835636.

[15] Shaghaghi, M.; Adve, R. S. Task selection and scheduling in multifunction multichannel radars. In *2017 IEEE Radar Conference (RadarConf)*, 2017, pp. 0969–0974, doi:10.1109/RADAR.2017.7944344.

[16] Shaghaghi, M.; Adve, R. S. Machine learning based cognitive radar resource management. In *2018 IEEE Radar Conference (RadarConf18)*, 2018, pp. 1433–1438, doi:10.1109/RADAR.2018.8378775.

[17] Briheche, Y.; Barbaresco, F.; Bennis, F.; et al. Theoretical complexity of grid cover problems used in radar applications. *Journal of Optimization Theory and Applications*, volume 179, no. 3, 2018: pp. 1086–1106.

[18] Deb, K.; Pratap, A.; Agarwal, S.; et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, volume 6, no. 2, 2002: pp. 182–197, doi:10.1109/4235.996017.

[19] Sun, J.; Yi, W.; Varshney, P. K.; et al. Resource Scheduling for Multi-Target Tracking in Multi-Radar Systems With Imperfect Detection. *IEEE Transactions on Signal Processing*, volume 70, 2022: pp. 3878–3893, doi:10.1109/TSP.2022.3191800.

[20] Zhang, H.; Liu, W.; Yang, X. Resource saving based dwell time allocation and detection threshold optimization in an asynchronous distributed phased array radar network. *Chinese Journal of Aeronautics*, volume 36, no. 11, 2023: pp. 311–327, ISSN 1000-9361, doi:https://doi.org/10.1016/j.cja.2023.06.017.

[21] Shi, C.; Ding, L.; Wang, F.; et al. Low Probability of Intercept-Based Collaborative Power and Bandwidth Allocation Strategy for Multi-Target Tracking in Distributed Radar Network System. *IEEE Sensors Journal*, volume 20, no. 12, 2020: pp. 6367–6377, doi:10.1109/JSEN.2020.2977328.

[22] Garey, M. R.; Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1979, ISBN 0716710447.

[23] Korte, B. H.; Vygen, J.; Korte, B.; et al. *Combinatorial optimization*, volume 1. Springer, 2011.

[24] Chvatal, V. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, volume 4, no. 3, 1979: pp. 233–235, ISSN 0364765X, 15265471. Available from: `http://www.jstor.org/stable/3689577`

[25] Dinur, I.; Steurer, D. Analytical approach to parallel repetition. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, 2014, pp. 624–633.

[26] Caprara, A.; Toth, P.; Fischetti, M. Algorithms for the set covering problem. *Annals of Operations Research*, volume 98, no. 1-4, 2000: pp. 353–371.

[27] Hoffman, A. J.; Kruskal, J. B. Integral boundary points of convex polyhedra. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, 2010: pp. 49–76.

[28] Schöbel, A. Set covering problems with consecutive ones property. Technical report.

[29] Kasirzadeh, A.; Saddoune, M.; Soumis, F. Airline crew scheduling: models, algorithms, and data sets. *EURO Journal on Transportation and Logistics*, volume 6, no. 2, 2017: pp. 111–137, ISSN 2192-4376, doi:https://doi.org/10.1007/s13676-015-0080-x. Available from: `https://www.sciencedirect.com/science/article/pii/S2192437620300820`

[30] Marchiori, E.; Steenbeek, A. An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. In *Workshops on Real-World Applications of Evolutionary Computation*, Springer, 2000, pp. 370–384.

[31] Farahani, R. Z.; Asgari, N.; Heidari, N.; et al. Covering problems in facility location: A review. *Computers & Industrial Engineering*, volume 62, no. 1, 2012: pp. 368–407, ISSN 0360-8352, doi:https://doi.org/10.1016/j.cie.2011.08.020. Available from: `https://www.sciencedirect.com/science/article/pii/S036083521100249X`

[32] Aktaş, E.; Özaydın, Ö.; Bozkaya, B.; et al. Optimizing fire station locations for the Istanbul metropolitan municipality. *Interfaces*, volume 43, no. 3, 2013: pp. 240–255.

[33] Hochbaum, D. S.; Levin, A. Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms. *Discrete Optimization*, volume 3, no. 4, 2006: pp. 327–340.

[34] Edwards, K.; Griffiths, S.; Kennedy, W. S. Partial interval set cover–trade-offs between scalability and optimality. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, Springer, 2013, pp. 110–125.

[35] Krupa R, D.; Basu Roy, A.; De, M.; et al. Demand hitting and covering of intervals. In *Algorithms and Discrete Applied Mathematics: Third International Conference, CALDAM 2017, Sancoale, Goa, India, February 16-18, 2017, Proceedings 3*, Springer, 2017, pp. 267–280.

[36] Butman, A.; Hermelin, D.; Lewenstein, M.; et al. Optimization problems in multiple-interval graphs. *ACM Transactions on Algorithms (TALG)*, volume 6, no. 2, 2010: pp. 1–18.

[37] Decan, A. portion: Python data structure and operations for intervals. Available from: `https://github.com/AlexandreDecan/portion`

[38] Bynum, M. L.; Hackebeil, G. A.; Hart, W. E.; et al. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, third edition, 2021.

[39] Hart, W. E.; Watson, J.-P.; Woodruff, D. L. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, volume 3, no. 3, 2011: pp. 219–260.

[40] Forrest, J.; Ralphs, T.; Vigerske, S.; et al. coin-or/Cbc: Release releases/2.10.11. Oct. 2023, doi:10.5281/zenodo.10041724. Available from: `https://doi.org/10.5281/zenodo.10041724`

[41] Harris, C. R.; Millman, K. J.; van der Walt, S. J.; et al. Array programming with NumPy. *Nature*, volume 585, no. 7825, Sept. 2020: pp. 357–362, doi:10.1038/s41586-020-2649-2. Available from: `https://doi.org/10.1038/s41586-020-2649-2`

[42] Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, volume 9, no. 3, 2007: pp. 90–95, doi:10.1109/MCSE.2007.55.

[43] Garrett, J. D. garrettj403/SciencePlots. Sept. 2021, doi:10.5281/zenodo.4106649. Available from: `http://doi.org/10.5281/zenodo.4106649`