Optimization of search patterns for fixed-panel tridimensional scanning radars

Yann Briheche

Thèse de Doctorat

Yann BRIHECHE

Mémoire présenté en vue de l'obtention
du grade de Docteur de l'Ecole Centrale de Nantes
Sous le label de l'UNIVERSITÉ BRETAGNE LOIRE

École doctorale : SPIGA

Discipline : Informatique Appliquées
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes

Soutenue le 30 Novembre 2017

Optimisation du maillage de la veille sur radar à balayage électronique à panneau fixe

JURY

Rapporteurs :	**FADEL Georges**, Prof., Clemson University **YANNOU Bernard,** Prof., CentraleSupélec
Présidents: Examinateurs:	**GRIFFITHS Hugh**, Prof., University College London **BHATTACHARJYA Rajib**, Prof., Indian Institute of Technology Guwahati **BARBARESCO Frederic**, Ing, Thales
Directeur de thèse :	**BENNIS Fouad**, Prof., Ecole Centrale de Nantes
Co-directeur de thèse :	**CHABLAT Damien**, Dr, CNRS

Contents

Introduction

Context

Classically, most people envision radars as they are often represented in cinema: a small round screen, circularly swept by a cone, displaying blinking points and beeping whenever a target is detected. That vision, which might have been true in the past, is no longer an accurate representation.

In the last decades, radar systems have become increasingly complex but also more versatile. Their tasks have extended accordingly with their capabilities. This evolution was greatly favoured by the development of electronics and digital processing throughout the modern industry. Modern radars are faster, adaptable and rely heavily on electronic systems. They can now dynamically sweep in random order their surroundings using electronic panel antennas, free from limitations of rotating antennas and sequential scanning. Modern radars incorporate digital high-rate reception processing chains, with high-performance numerical processors relying on advanced statistical estimators.

This paradigm shift results in a fundamentally different underlying mathematical model for radar engineering. The integration of this evolution to the engineering methodology will permit to harness the full potential of modern radar systems.

And this evolution also impact how radar are used; while traditional radars were each dedicated to a single task, modern radars are now multifunction as they use their new-found flexibility to perform multiple tasks in parallel: scanning, tracking, identification, communication, clutter mapping, etc. Each of those tasks requires time for emission, propagation, reception and treatment of the radar signal. Radar time is the essential resource in radar tasks management.

In modern warfare, increasingly intelligent and adaptable systems compete against each other, seeking reactivity in increasingly shorter time and managing ever more information. In this context, optimizing radar efficiency is necessary to achieve desired performances in due time and avoid overload.

Motivation and Objectives

One the main challenges for radar engineering is to appropriate modern tools to make proficient use the increased computational power of the digital era: mathematical modelling, algorithmics, operational research and optimization.

Those techniques are used with two objectives in mind: the production of aided-design tools, to facilitate, improve and speed up design and simulation of radar architectures; and the development of real-time practical algorithm for optimizing resource management and radar processing in operational situation.

One particular radar task, fundamental but costly is *searching* (or *scanning*), of yet-unknown targets. Radar search optimization is a main topic for radar resource management, and the subject of this thesis, a joint project between THALES AIR SYSTEMS, the *Direction Générale de l'Armement* (DGA) of the French Ministry of Defence and the Laboratory of Digital Sciences of Nantes (*Laboratoire des Sciences du Numérique de Nantes*). The thesis main objectives are:

- to define the theoretical framework and mathematical model of radar search optimization for three-dimensional scanning radars.
- to identify, implements and test the appropriate approaches and algorithms for solving radar search optimization problems.

The work accomplished during the thesis is:

- a general problem formulation for radar search pattern optimization of phased-array antenna radar. This formulation can also be extended to any radar capable of dynamical beamforming, i.e. electronic control of the antenna radiation pattern.
- a procedure for approximating this problem as a combinatorial *cover problem*, and solving the problem using integer programming methods.
- extensions of the initial formulation accounting for localized clutter, terrain masking, localized scan update rate and multi-mission constraints.
- computational improvements based on reduction methods for decreasing the number of variables and/or constraints, and thus the size, of the combinatorial problem.

- explore and formulate future research leads, such as how to exploit overlaps in the radar search pattern, formulated as a probability cover problem.
- implementation of a software framework for optimization of radar search patterns, identification of short-term applications in aided-design and performance simulations; and long-term applications in real-time radar resource management.

Thesis outline

This thesis contents are organized as follows. Chapter 1 presents the basic principles of radar theory and builds the mathematical radar model which will be considered in the rest of the thesis. Chapter 2 focuses on optimization and complexity theory, presents the theoretical framework for solving combinatorial cover problems as well as results on the computational complexity of radar cover problems. Chapter 3 defines the general formulation for radar search optimization, and describes a procedure for its approximation and solving as a combinatorial cover problem. Chapter 4 presents extensions for integrating localized multi-mission constraints, computational improvements for faster computation, and explore future research leads of interests.

The thesis concludes on synthesis of the work achieved, the possible applications and the continuation of this research.

Chapter 1

Radar theory and mathematical model

1.1 History

The term RADAR is the contraction of *"RAdio Detection And Ranging"*. It encompasses all systems and techniques for detecting and analysing distant objects through the use of radio waves, which usually refers to electromagnetic waves with frequencies between a few kilohertz to several hundred gigahertz.

The first radar experiments were pioneered by German physicist Heinrich Hertz in the late 19th century, applying James Maxwell's ideas. However, radar technology has most significantly developed during the last decades, principally for military use and defence applications.

Radars are nowadays essential assets in modern warfare and military defense, ever since WWII. They also play an important role in civilian applications, most notably in flight control with the ever increasing traffic, but also in weather forecasting, topography and geology. Radar research has been prolific in the latter part of the 20th century during which many radar systems and technological improvements have been made.

Radar theory covers a wide variety of fields, from antenna design, focusing on the electromagnetic properties of radiating elements; to signal processing, studying the structure and efficiency of transmitted signals; to statistics for extracting reliable information for target detection and analysis.

1.2 Radar basic principle

A radar system detects an object by propagating electromagnetic waves, from which it can also infer information regarding the object. This process can be divided into three steps :

- The radar first sends an electromagnetic wave in the direction of observation.
- Upon encountering an object, the wave is reflected and partially propagates back to the radar antenna.
- The radar receives and processes the reflected wave to detect an object and estimate its characteristics, for example position or speed.

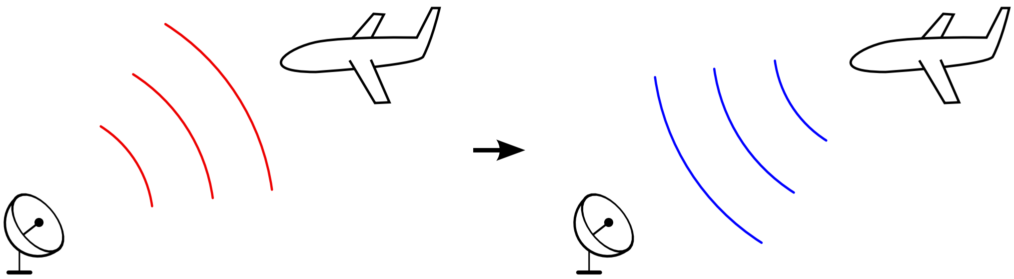

Figure 1.1: Radar emission and reception

Unfortunately, the received signal is polluted with environmental noise. The further the object is, the weaker the echo is and the harder it becomes to distinguish the echo from noise. Detection of weak echo signals can be improved through different parameters:

- The emitter antenna power. The most straightforward solution, but with significant material, logistic and energetic costs. A more powerful antenna will be bigger, and use more energy, thus producing more heat and requiring a better cooling system. This is usually not the preferred solution, rather used as a last resort.
- The antenna gain, i.e. the radar capacity to focus most of its energy in the same direction rather than dissipate it in all directions. Concentrating the radiating power decreases the detection angular area but improves the detection range. Modern radars rely on electronics to numerically control and dynamically generate a desired radiation pattern.

- The radar signal waveform, i.e. the shape of the emitted signal. After reflection, a longer echo is easier to extract from noise, as the echo has a consistent temporal structure. The longer it is, and the more it contrasts with the randomness of noise, typically assumed *white* (i.e. incoherent between any two instants). A longer signal means sending more energy on the target. Time *integration* of the signal summed this energy. Integration is constructive for echoes but destructive for noise.

The formal mathematical relation between those parameters and the detection range is called the *radar equation.*

1.3 Radar equation

1.3.1 Definition

One way to express the radar equation is the relationship between the energy reflected by a target to the radar and characteristics of the radar (average emitted power, antenna gain), of the target (radar cross-section, distance to the radar) and various losses (system, scanned).

There are however numerous formulations for the radar equation, depending on configuration and model, which are all mathematically equivalent. The formulation depends on the setting: to design and size the required system and parameters to achieve desired detection (for given target at given range, etc.), or conversely to compute the performances of a given radar architecture. The equation always models the same phenomenon and quantify the propagation and dispersion of radar waves travelling forth and back between the radar and a target:

$$E_r = \frac{P\,T\,g_t\,g_r\,\lambda^2\,\sigma}{(4\pi)^3\,R^4\,L_t} \tag{1.1}$$

with :

- P the antenna average power (W),
- g_t (g_r) the antenna emission (reception) gain in the target direction (dB),
- T the emitted signal time duration (s),
- λ the signal wavelength (m),
- σ the target radar cross-section, its “visibility” to the radar (m^2),

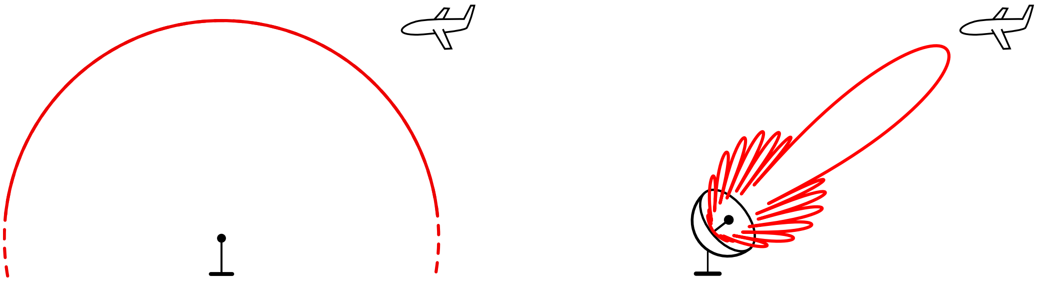

Figure 1.2: Isotropic antenna (left) and directive antenna (right)

- R the radar-target distance (m),
- L_t the total losses (encompassing propagation, scanned and internal losses) (dB).

1.3.2 Energetic dispersion interpretation

The radar equation can be explained by a relatively simple physical interpretation of waves propagation. Consider the radar antenna as a point source in the *far-field hypothesis* (i.e. "seen from far away"). The antenna is isotropic if it emits the same power in all directions, and has a constant gain. It is directive if the antenna focuses the power in certain directions, and has a variable gain. Both cases are showcased in Figure 1.2.

An isotropic antenna radiates its power P uniformly emitting spherical waves at far-field. At a distance R from the radar, its power is distributed evenly on a sphere with a surface $4\pi R^2$, as shown in Figure 1.3. For a directive antenna, the power distribution is proportionate to the antenna gain. The power flux density radiating from the antenna is

$$\frac{Pg_t}{4\pi R^2}$$

A target with radar cross-section σ at range R will partially intercept and reflect this power. The target can be viewed as a point source, re-emitting spherical waves. The reflected power at a distance R from the target is similarly distributed on the sphere with radius R, as shown in Figure 1.3. The reflected power flux density intercepted by the radar antenna is

$$\frac{Pg_t}{4\pi R^2}\frac{\sigma}{4\pi R^2}$$

and will be intercepted by the antenna effective reception area $A_e = g_r\lambda^2/4\pi$ [1], during the entire duration T of the signal. The total energy reflected to

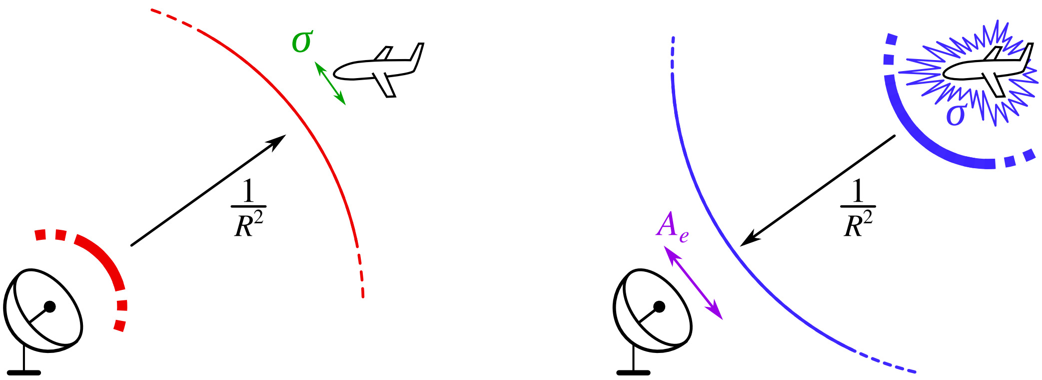

Figure 1.3: Energetic propagation and reflection of a radar signal

the radar is

$$\frac{Pg_t}{4\pi R^2}\frac{\sigma}{4\pi R^2}\frac{g_r\lambda^2}{4\pi}T$$

accounting for losses L, which results in the radar equation (1.1).

1.4 Radiation pattern model

1.4.1 Phased array

The radar antenna model is a bi-dimensional phased-array of K-by-L evenly spaced radiating elements, shown in Figure 1.4, with horizontal spacing d_x and vertical spacing d_z. In the array local Cartesian coordinates system $Oxyz$, the position of radiating element (k,l) is given by

$$\vec{\mathbf{p}}_{k,l} = (x,y,z) = (-ld_h, 0, kd_v)_{xyz}$$

Each radiating element is an isotropic electromagnetic source, whose phase and amplitude can be freely controlled

$$s_{k,l}(t) = A_{k,l}\, e^{j\phi_{k,l}}\, s(t)$$

with the amplitude $A_{k,l} \in [0,1]$ and the phase $\phi_{k,l} \in [0,2\pi[$ of radiating element $(k,l) \in \{0,...,K-1\} \times \{0,...,L-1\}$, and the emission signal $s(t)$ feed in the antenna.

A phase-amplitude illumination law of the antenna array is defined by a set of values $\{a_{k,l}\}$ in the complex open unit disk $\mathbb{D}$:

$$\{a_{k,l} = A_{k,l}\, e^{j\phi_{k,l}} \in \mathbb{D} : 0 \leq k < K, 0 \leq l < L\}$$

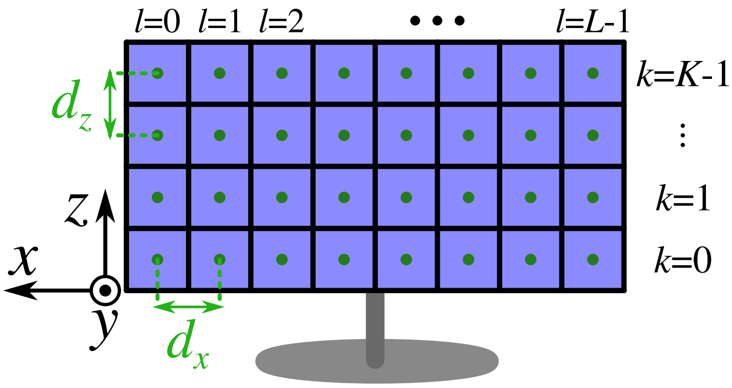

Figure 1.4: Phased array local coordinates system. Green dots are geometric centers of radiating elements

A direction of observation is defined by the antenna local spherical coordinates $(\phi, \theta) \in [0, \pi]^2$, see Figure 1.5. The associated unitary direction vector is defined in Cartesian coordinates

$$\vec{\mathbf{u}} = (u, v, w) = (\cos(\theta)\sin(\phi), \sin(\theta), \cos(\theta)\cos(\phi))_{uvw} \tag{1.2}$$

known as *direction cosines*. It is common in practice to use only (u, v), as w is immediately deduced by $u^2 + v^2 + w^2 = 1$ and $w \geq 0$. Remark that the array local coordinates (x, y, z) and direction cosines coordinates (u, v, w) are different coordinate systems, related by the following relations

$$\begin{cases} u &= -x \\ v &= z \\ w &= y \end{cases} \quad \text{or} \quad \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

and the radiating element (k, l) position can be written in direction cosines coordinates as

$$\vec{\mathbf{p}}_{k,l} = (ld_h, kd_v, 0)_{uvw}$$

1.4.2 Beamforming emission

For a far-field target in direction $\vec{\mathbf{u}}$, the array antenna can be approximated as a sum of point sources. The emitted signal is the aggregation of each source signal on a wavefront, which is perpendicular to the direction $\vec{\mathbf{u}}$.

When the wavefront is not coplanar to the antenna plane, a phase shift appear among the signals. The phase shift between element (k, l) and a reference element $(0, 0)$ can be geometrically expressed as an optical pathway shift

$$\delta_{k,l} = \vec{\mathbf{p}}_{k,l} \cdot \vec{\mathbf{u}} = ld_h u + kd_v v$$

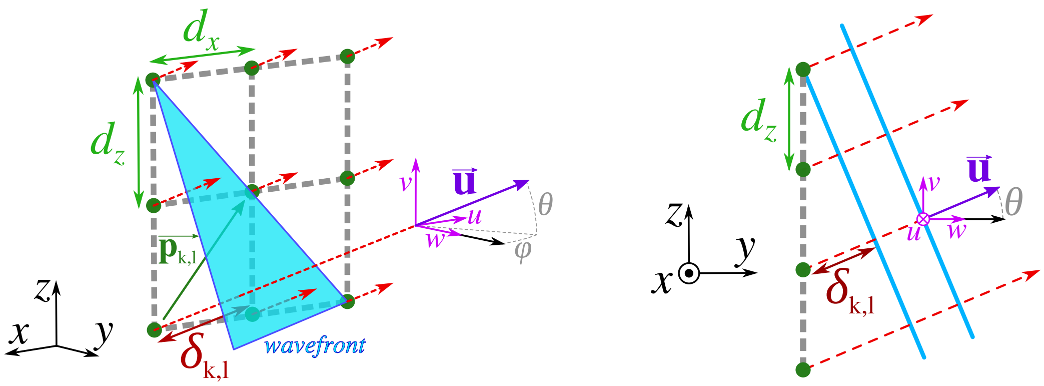

Figure 1.5: Wavefront propagation and direction cosines coordinates

as shown in Figure 1.5. The total emitted signal can be expressed as

$$\sum_{k=0}^{K-1}\sum_{l=0}^{L-1} a_{k,l}\ s(t)\ e^{j2\pi\frac{\delta_{k,l}}{\lambda}} = \left(\sum_{k=0}^{K-1}\sum_{l=0}^{L-1} a_{k,l}\ e^{j2\pi\frac{ld_h u+kd_v v}{\lambda}}\right) s(t) = g_e(u,v)\ s(t)$$

with λ the signal carrier wavelength, and $g_e(u,v)$ the *emission gain* of the antenna in direction $\vec{\mathbf{u}}$, also called *array factor* [2, 3]:

$$g_e(u,v) = \sum_{k=0}^{K-1}\sum_{l=0}^{L-1} a_{k,l}\, e^{j2\pi\frac{ld_x u+kd_y v}{\lambda}} = \sum_{k=0}^{K-1}\sum_{l=0}^{L-1} A_{k,l}\, e^{j\phi_{k,l}}\, e^{j2\pi\frac{ld_x u+kd_y v}{\lambda}} \tag{1.3}$$

Remark the origin choice only impacts the global phase of the radiation pattern, but not the phase shifts between elements, nor the absolute value of the radiation pattern.

In Equation (1.3), the emission gain corresponds to the discrete bi-dimensional Fourier transform of the phased-array illumination law $\{a_{k,l}\}$ with substitution $(\nu,\mu) = (\frac{d_x}{\lambda}u, \frac{d_y}{\lambda}v)$

$$g_e(u,v) = \sum_{k=0}^{K-1}\sum_{l=0}^{L-1} a_{k,l}\, e^{j2\pi(l\nu+k\mu)} = DFT(\{a_{k,l}\})(\nu,\mu)$$

Control of phases and amplitudes of the array elements can be used to shape the radiation pattern, relying on known principles of spectral signal processing. Windowing of the illumination law amplitude controls the pattern beam-shape, whereas a (u,v)-linear phase term added in the illumination law translates the radiation pattern in direction cosines spaces. This technique for controlling the radiation pattern is called *beamforming*. When used for translating the radiation pattern main lobe through phase shifts, this technique is also called *beam-steering*. Beamforming is showcased in Figures 1.6 and 1.7.

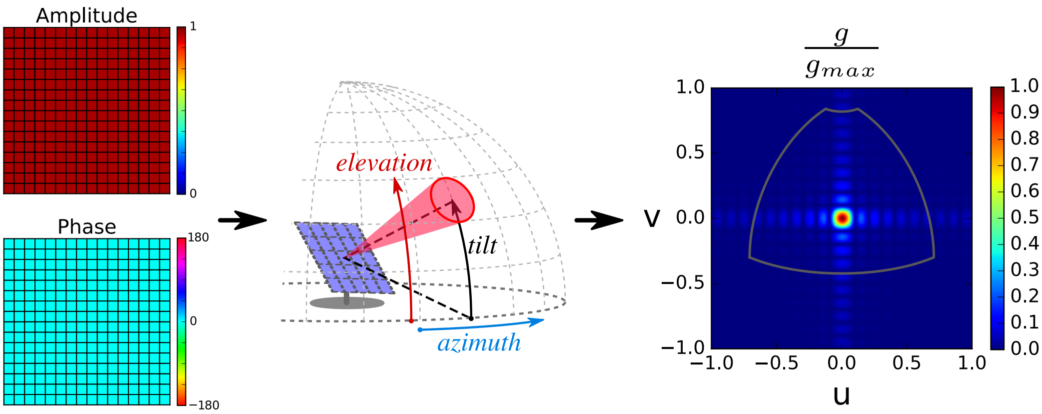

Figure 1.6: Centered narrow beam radiation pattern (middle, right) obtained with null-phase constant-amplitude illumination law (left) for the phased array

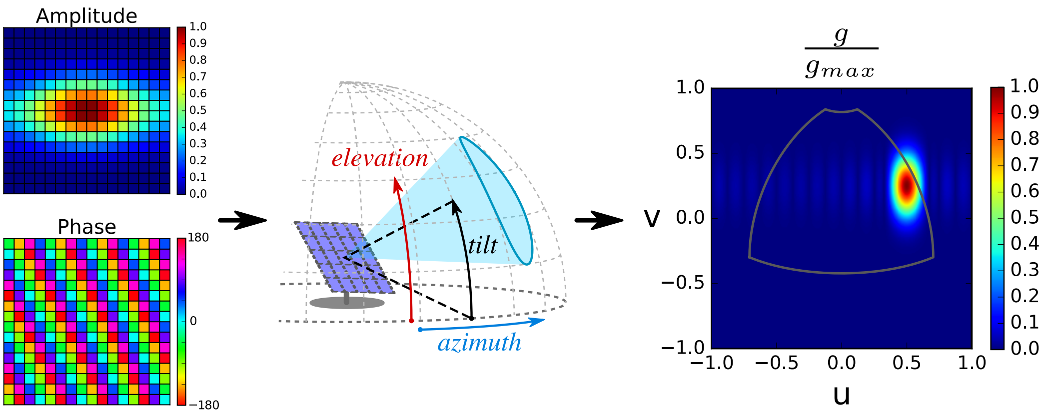

Figure 1.7: Steered widened beam radiation pattern (middle, right) obtained with linear-phased windowed-amplitude illumination law (left) for the phased array.

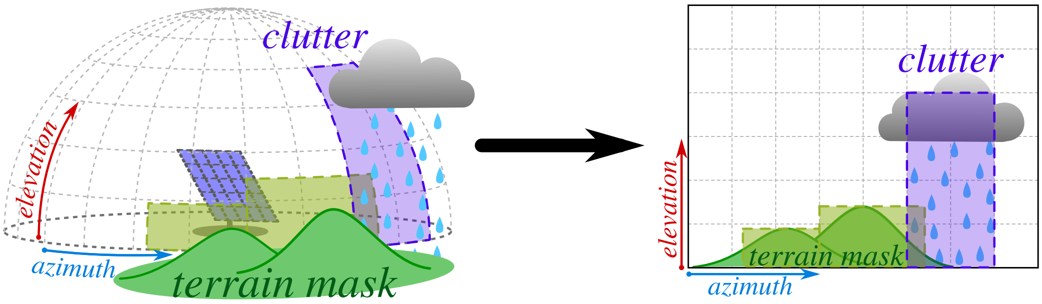

Figure 1.8: Operational situation with clutter (rain) and terrains masks, represented in azimuth-elevation coordinates.

1.4.3 Operational coordinates system and scanned losses

In operational situation, as shown in Figure 1.8, constraints and detection requirements are usually defined in a spherical coordinate systems bound to the local tangent plane of the Earth: the *azimuth-elevation* coordinates system with $(az, el) \in [0, \pi]^2$.

In the case where the antenna perpendicular Ox is co-linear the azimuth-elevation origin, the operational coordinates and the local antenna spherical coordinates are fused: $(az, el) = (\phi, \theta)$.

In practice, the antenna normal is tilted upwards by a tilt angle $t \in [0, \frac{\pi}{2}]$ as shown in Figure 1.6, to better center the radar emission space, the half-space $y > 0$, with the surveillance space, for which elevation is often positive, as there is often no point in emitting below the horizon.

Tilting the radar geometrically correspond to applying a rotation matrix with axis Ox and angle t to the antenna coordinates system, yielding the following relations between operational coordinates and antenna local direction cosines

$$\begin{array}{rcl} u & = & \cos(el)\sin(az) \\ v & = & \sin(el)\cos(t) - \sin(t)\cos(az)\cos(el) \\ w & = & \sin(el)\sin(t) + \cos(t)\cos(az)\cos(el) \end{array} \tag{1.4}$$

Reciprocal formulas can be obtained by inverting the previous equations

$$\begin{array}{rcl} az & = & \text{atan2}\left(u, \cos(t)w - \sin(t)v\right) \\ el & = & \text{asin}\left(\sin(t)w + \cos(t)v\right) \end{array} \tag{1.5}$$

Substitution between coordinates systems is easily done using Equations (1.4) and (1.5). In the following, all functions can indiscriminately switch between parameters (az, el) and (u, v).

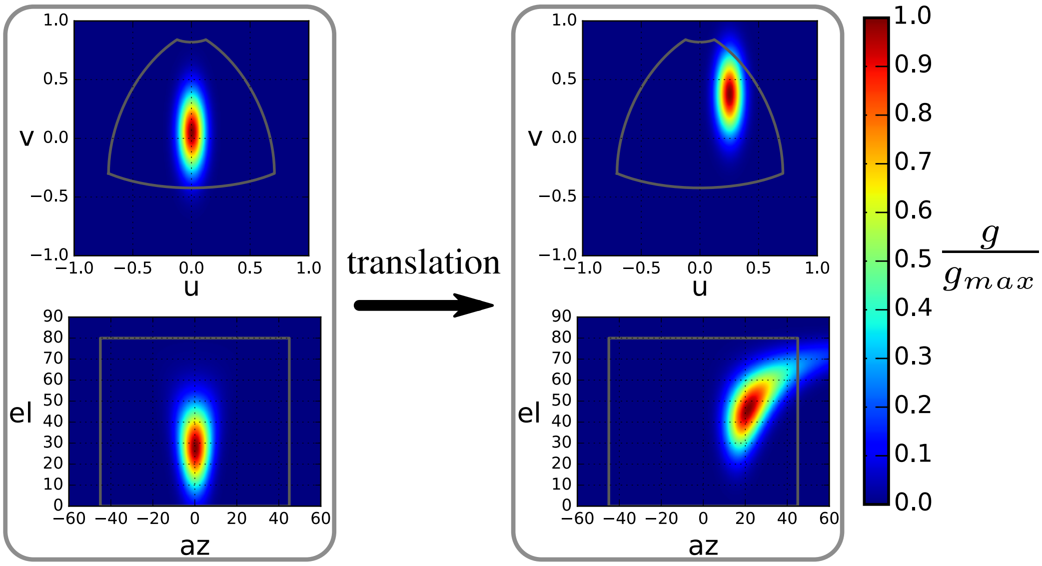

Figure 1.9: Distortion between direction cosines (top) and operational coordinates (bottom) after translation of the beam

In the more general setting, where the two coordinates systems are completely unbound, three rotations are required to switch between them, with two additional angles known as *bearing* and *queer*. Equations (1.4) and (1.5) can be generalized to account the three rotations, see Annexe 1.

A peculiar property of those coordinate substitutions is that they do not preserve areas. Informally, substitution of direction cosines by operational coordinates "spreads" surfaces in a non-uniform fashion [4]. So while translating a beam-shaped radiation away from the array perpendicular direction (via a linear phase term in the array illumination law) preserves its area in direction cosines space, the same beam becomes distorted in operational coordinates, see Figure 1.9. It cover a larger solid angle but with weaker angular power density, resulting in anisotropic *scanned losses*:

$$L_s = \cos(\delta)^{-1}$$

where δ is the angle between the antenna array perpendicular direction and the direction of observation.

Mathematically, the scanned loss factor is the dilatation ratio between an infinitesimal solid angle element in operational coordinates

$$\mathrm{d}\Omega = \cos(el)\,\mathrm{d}az\,\mathrm{d}el$$

and an infinitesimal surface element in direction cosines space

$$\mathrm{d}u\,\mathrm{d}v$$

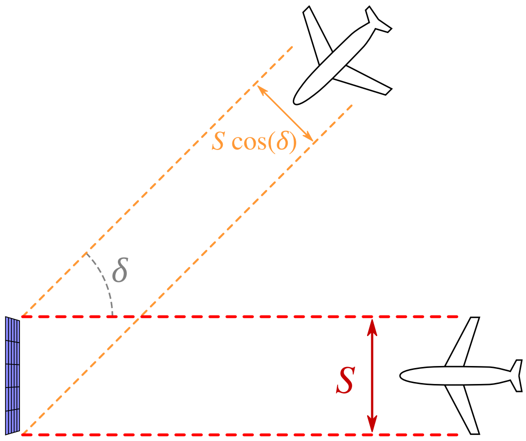

Figure 1.10: Physical interpretation of scanned losses

The scanned loss factor can be computed from the Jacobian matrix J_F of function $F : (az, el) \to (u, v)$ defined by Equation (1.4):

$$\begin{aligned}
\mathrm{d}u\,\mathrm{d}v &= |\det(J_F(az, el))|\,\mathrm{d}az\,\mathrm{d}el \\
&= |\sin(el)\sin(t) + \cos(az)\cos(el)\cos(t)|\cos(el)\,\mathrm{d}az\,\mathrm{d}el \\
&= |\sin(el)\sin(t) + \cos(az)\cos(el)\cos(t)|\mathrm{d}\Omega \\
&= w\,\mathrm{d}\Omega
\end{aligned}$$

and the scanned loss is equal to the third direct cosine coordinate $w \in [0, 1]$:

$$L_s = \frac{\mathrm{d}\Omega}{\mathrm{d}u\,\mathrm{d}v} = \frac{1}{w} = \frac{1}{\cos(\delta)} = \frac{1}{\sin(el)\sin(t) + \cos(az)\cos(el)\cos(t)} \tag{1.6}$$

with δ between vector $\vec{\mathbf{u}}$ pointing the direction of observation and the array antenna normal unit vector $\vec{\mathbf{n}}$. Scanned losses do not occur in the direction perpendicular to the antenna, and increase as the direction of observation from the antenna perpendicular. Scanned losses also occur twice, at emission and at reception, and are squared in the radar equation.

A physical interpretation of scanned losses is shown in Figure 1.10: $w = \vec{\mathbf{u}} \cdot \vec{\mathbf{n}} = \cos(\delta)$ is also the ratio between the apparent surface of the antenna and its real surface, from a target in direction $\vec{\mathbf{u}}$. At emission, the target “sees” a smaller antenna, and receives a proportionally decreased angular power density. Similarly at reception, the “effective” area of the antenna receiving the reflected energy is smaller.

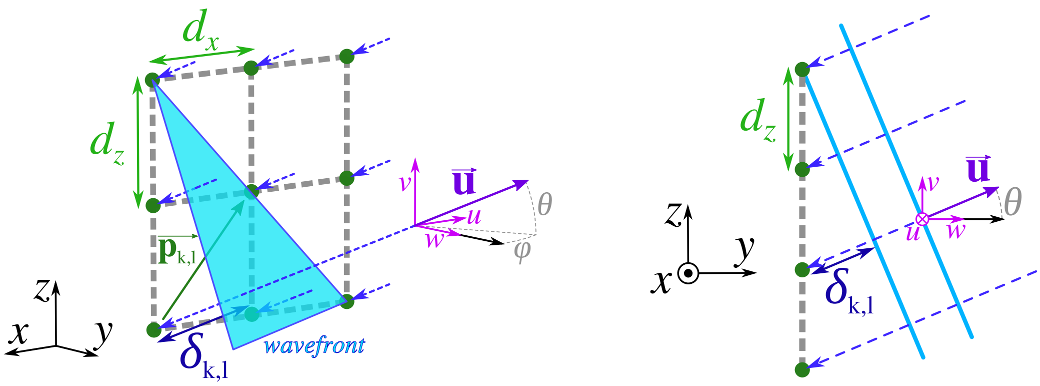

Figure 1.11: Wavefront reception

1.4.4 Digital beamforming reception

According to Fermat's principle of light least travel time, optical pathways are reversed between emission and reception. A reception phased-array antenna model (which may be the same antenna used for emission) has similarities with the emission antenna model described previously.

If the antenna is receiving a signal $r(t)$ from a far-field source, for example a reflecting target, located in direction pointed by $\mathbf{u}$, then the wavefront of the received signal is orthogonal to $\mathbf{u}$ as shown in 1.11. The signal received by each source is

$$r_{k,l} = r(t)e^{j2\pi\frac{\delta_{k,l}}{\lambda}}$$

with an optical path shift $\delta_{k,l} = \vec{\mathbf{p}}_{k,l} \cdot \vec{\mathbf{u}}$.

By controlling phase $\phi_{k,l}$ and amplitude $A_{k,l}$ of the received signals and aggregating their values, the target signal can be amplified:

$$\sum_{k=0}^{K-1}\sum_{l=0}^{L-1} a_{k,l} \left(r(t)\ e^{j2\pi\frac{\delta_{k,l}}{\lambda}}\right) = \left(\sum_{k=0}^{K-1}\sum_{l=0}^{L-1} a_{k,l}\ e^{j2\pi\frac{ld_h u + kd_v v}{\lambda}}\right) r(t) = g_r(u,v)\ r(t)$$

with λ the signal carrier wavelength, and $g_r(u,v)$ the *reception gain* of the antenna in direction $\vec{\mathbf{u}}$

$$g_r(u,v) = \sum_{k=0}^{K-1}\sum_{l=0}^{L-1} A_{k,l}\, e^{j\phi_{k,l}}\, e^{j2\pi\frac{ld_x u + kd_y v}{\lambda}}$$

which is, as expected, the same formula than the emission gain. A physical interpretation of this result is to view the reception gain as beamforming a *reception pattern*, with similar properties than radiation pattern for emission.

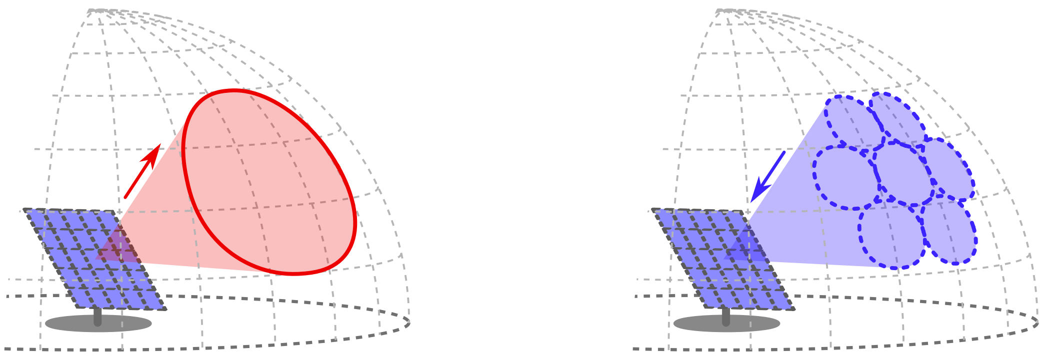

Figure 1.12: Emission beamforming (left) and reception digital beamforming (right)

A key difference between emission and reception is that choice of amplitude-phase reception law $\{a_{k,l}\}$ does not have to be the same than the illumination law at emission. In fact, since the radar directly receives the signals $r_{k,l}$ in each element, it is possible to immediately digitize those signals out of the array and compute multiple digital reception patterns in parallel. This technique is called *digital beamforming* and is illustrated in Figure 1.12.

With this approach, it is possible to scan a wide area using multiple narrow (thus more energetically powerful) beam. The limit of digital beamforming depends on two parameters:

- the narrow beam width of the radar, which is inversely proportional to its antenna surface area.
- the digital processor capacity, which limits how many beam-forming computations can be performed in parallel.

The narrow beam radiation pattern is generate by a constant amplitude illumination law (i.e $\forall(k,l), |a_{k,l}| = 1$), like the centered narrow beam displayed in Figure 1.6. Any narrow beam is a translation of the centered narrow beam by using a linear phase term in illumination law, and has the same width in direction cosines coordinates, but not in operational coordinates, where scanned distortions occur. The absolute value of the reception gain of the centered narrow beam is

$$|g_r(u,v)| = \frac{\sin\left(\frac{\pi K d_x}{\lambda}u\right)}{\sin\left(\frac{\pi d_x}{\lambda}u\right)} \frac{\sin\left(\frac{\pi L d_z}{\lambda}v\right)}{\sin\left(\frac{\pi d_z}{\lambda}v\right)}$$

with its maximum value at the center being $|g_r(0,0)| = KL$. The half-power beam-width of the radiation pattern can be approximated as an ellipse with semi axis $\Delta u_{3dB} = 2u_0$ and $\Delta v_{3dB} = 2v_0$, where u_0 and v_0 are solutions of the system

$$\begin{cases} \frac{\sin\left(\frac{\pi K d_x}{\lambda}u\right)}{\sin\left(\frac{\pi d_x}{\lambda}u\right)} & = \sqrt{\frac{1}{2}}K \quad \Leftrightarrow \sqrt{2}\sin\left(\frac{\pi K d_x}{\lambda}u\right) & = K\sin\left(\frac{\pi d_x}{\lambda}u\right), \quad 0 < u < \frac{\lambda}{2Kd_x} \\ \frac{\sin\left(\frac{\pi L d_z}{\lambda}v\right)}{\sin\left(\frac{\pi d_z}{\lambda}v\right)} & = \sqrt{\frac{1}{2}}L \quad \Leftrightarrow \sqrt{2}\sin\left(\frac{\pi L d_z}{\lambda}v\right) & = L\sin\left(\frac{\pi d_z}{\lambda}v\right), \quad 0 < v < \frac{\lambda}{2Ld_z} \end{cases}$$

which can numerically be solved by using root-finding line search, such as the popular Brent's method [5] (implemented in MATLAB by `fsolve`, and in SciPy by `scipy.optimize.brentq`). The half-power narrow beamwidth can also be approximated using

$$\begin{cases} \Delta u_{3dB} \approx 0.89\frac{\lambda}{\pi K d_x} & \text{if } K \gg 1 \\ \Delta v_{3dB} \approx 0.89\frac{\lambda}{\pi L d_z} & \text{if } L \gg 1 \end{cases}$$

which are the formulas for a continuous rectangular electromagnetic source. Physically, a discrete array with enough elements can be viewed as a continuous source.

The half-power beamwidth of the centered narrow beam is approximately the area $A_{3db} = 2\pi\Delta u_{3dB}\Delta v_{3dB}$ of the ellipse with semi axis Δu_{3dB} and Δv_{3dB}.

Considering the number of parallel beamforming computations the digital processor can perform is a known system value $N_{DBF} \in \mathbb{N}$, the maximum area in direction cosines which can be scanned at reception is

$$A_{\max} = N_{DBF}\ A_{3db} = N_{DBF}\ 2\pi\Delta u_{3dB}\Delta v_{3dB}$$

and the minimum reception gain of digital beamforming is at most 3 decibels below the maximum gain of the antenna array

$$g_{DBF} = \sqrt{\frac{1}{2}KL}$$

1.5 Waveform model

1.5.1 Waveform definition and detection principle

The *waveform* is the shape along time of the signal emitted by the radar. The principle of radar detection is to "search" and try to "recognize" the waveform, the emitted signal shape, inside the received signal to find an echo reflected by a target, see Figure 1.13 for a simplified example.

The radar model in this thesis is a mono-static pulse-Doppler radar. A mono-static radar uses the same antenna for emission and reception, and

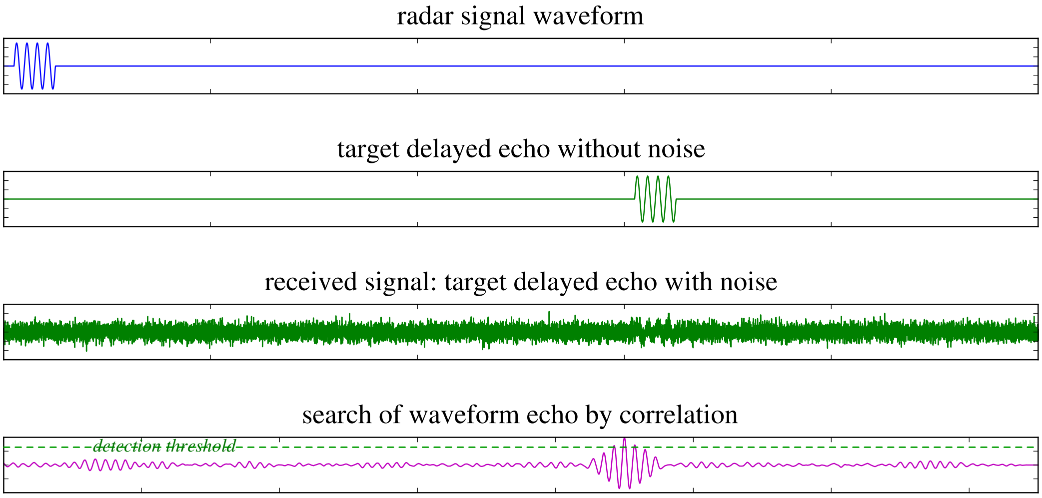

Figure 1.13: Research of a target echo of the waveform in the received signal

thus cannot receive while emitting. The waveform model is a series of short pulses (emission) alternating with silences (for reception). Those series of pulses are combined to increase the signal-to-noise ratio. This technique, used for improving detection, is called *integration*.

The waveform model presented in this thesis is energetic, and does not details the signal processing aspects of waveform design: pulse modulation, spectral occupation, ambiguity function, encoding, etc., nor the associated processing chain: demodulation, matched/mismatched filtering, etc.

Inside a waveform, series of pulses with similar characteristics are grouped together, such a group is called a *burst*. A waveform is thus a series of bursts, and each burst is a series of pulses, as shown in Figure 1.14. The signal parameters are different from burst-to-burst inside a waveform, but are constant inside a burst:

- τ : pulse width (s).
- T_p : pulse repetition interval, the period between the start of two successive pulses (s), thus $T_p - \tau$ is the silence duration between a pulse end and the next pulse start.
- N_p : number of pulses in the burst, with the burst duration being $N_p T_p$
- f : duty cycle, ratio between the pulse width and the pulse repetition interval
$$f = \frac{\tau}{T_p}$$

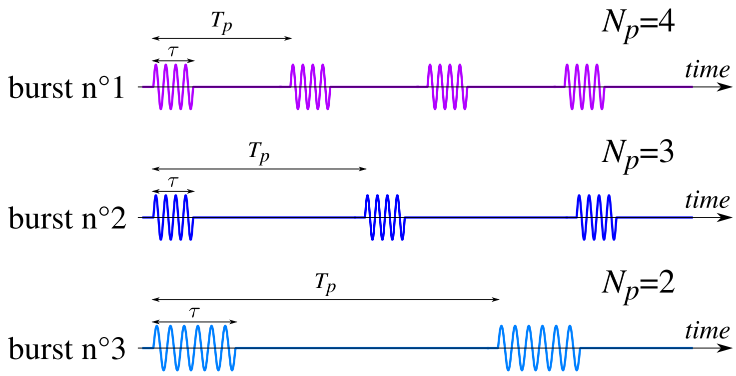

Figure 1.14: Waveform structure decomposition

which also relates the radar average power P_m to the radar peak power P_m

$$P_m = P_p f$$

and the total energy emitted during the waveform is $P_m T$ where T is the waveform total duration.

In presence of target, the emitted signal is reflected back toward the radar. A target at range R results in a pulse echo with a time delay

$$\Delta t = \frac{2R}{c}$$

where c the speed of light, since the signal takes Δt to travel the radar-target distance R forth and back at speed c. If the target has a radial speed v, then between two pulses the target gets closer by $2vT_p \ll R$ and the received signal has a phase shift

$$\Delta\phi = 2\pi\frac{2vT_p}{\lambda} = \frac{4\pi vT_p}{\lambda}$$

with λ the signal carrier wavelength. Both distance are shown on Figure 1.15.

Each pulse is individually too weak to allow detection. However, under the assumption of *white* noise, which implies that noise is independent between any two instants along the time axis, it is possible to combine several impulsions to improve detection. This approach is called *integration*.

Most integration schemes fall under two categories: coherent integration, which makes use of amplitude and phase information of the signal, and incoherent integration where only amplitude is used, whereas phase is considered to be random (and thus “incoherent”) between impulsions. In the case of

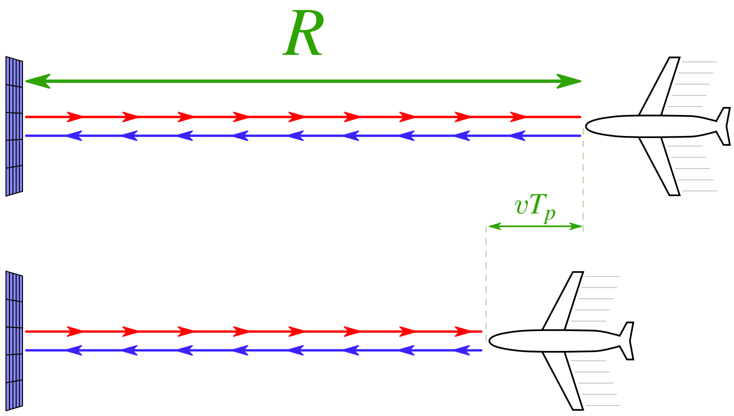

Figure 1.15: First pulse propagation (top) and second pulse propagation (bottom)

coherent integration, using Doppler filter-bank permits estimation of target radial speed, in addition to distance.

The principle of radar detection is to perform a hypothesis test on whether the received signal contains a waveform echo from a target at a given range with a given radial speed. For each range-speed hypothesis, an estimator is computed from the received signal, testing the presence of an echo with time delay Δt and phase shift $\Delta\phi$. The detection hypothesis (i.e. "a target is present at given range with given speed") is validated if the estimator is above a chosen detection threshold and rejected if it is below, see Figure 1.13. The radar system tests multiple combinations of range-speed hypotheses for which a target must detected. The number of tests depends on the limits and resolution of range and speed.

The detection probability that a target is correctly detected is P_d and the probability that a target is missed is $1 - P_d$ (type II error, known as "false negative"). The false alarm probability that a target is incorrectly detected from pure noise, when there is in fact no target, is P_{fa} (type I error, known as "false positive"). Obviously, a higher signal-to-noise ratio (i.e. power of a target echo relatively to noise level) improves estimation and diminishes missed targets and false alarms.

In practice, the detection threshold is set to ensure a certain false alarm rate, as too frequent false alarms will mask true targets. The detection threshold depends only on noise parameters, and not on the target characteristics. For a square law detector, the normalized detection threshold is given by a formula [6, 7]

$$t = -\ln(P_{fa})$$

In performance measurements, Swerling models are often used to statistically represent reflecting properties and variability of generic targets [8]. For each Swerling model, the detection probability P_d can be computed depending on signal-to-noise ratio s and desired false alarm probability P_{fa} [6, 9]:

$$\begin{array}{ll} \text{Swerling I/II :} & P_d = {P_{fa}}^{\frac{1}{1+s}} \\ \text{Swerling III/IV :} & P_d = {P_{fa}}^{\frac{2}{2+s}} \left(1 - \frac{2s\ln(P_{fa})}{(2+s)^2}\right) \\ \text{Swerling 0 (V) :} & P_d = \int_{-\ln(P_{fa})}^{+\infty} e^{-(x+s)} \frac{1}{2\pi} \int_0^{2\pi} e^{2\sqrt{sx}\cos\theta} \mathrm{d}\theta \mathrm{d}x \end{array} \tag{1.7}$$

Reciprocally, knowing the desired detection and false alarm probabilities for a given target model, it is possible to numerically compute the minimum signal-to-noise ratio for achieving detection requirements.

1.5.2 Energetic model

Since a waveform is formally defined as a collection of bursts, its parameters are the aggregation of all its bursts parameters. A signal processing model of waveform and the corresponding radar processing chain fall outside the scope of this thesis. But a simpler energetic model of the waveform can be defined using fewer parameters, such that for a waveform w:

- T_w : the waveform total duration (s)
- f_w: the (average) dutycycle in the waveform.
- $s_w(P_d, P_{fa})$: the required signal-to-noise ratio in order to achieve detection and false alarm probabilities P_d, P_{fa}.

For a real system, the required signal-to-noise ratio s_w can be measured and stored in a database. With this approach, a system database of available waveforms with known performances in various scenarios can be computed. Another approach is to simulate waveform performances. A simple energetic model for doing so is described below.

The model uses Doppler filtering for pulse integration inside each burst; then performs double threshold detection to aggregate multiple bursts inside a waveform:

- *Pulse integration*: Doppler filtering is coherent integration, and N_p coherently integrated pulses can be viewed as one virtual pulse with an N_p-times stronger signal-to-noise ratio. Sterling mono-pulse Formulas (1.7) can be used to compute the required signal-to-noise ratio s_w for achieving burst detection probability $P_{d,b}$ and burst false alarm probability $P_{fa,b}$.

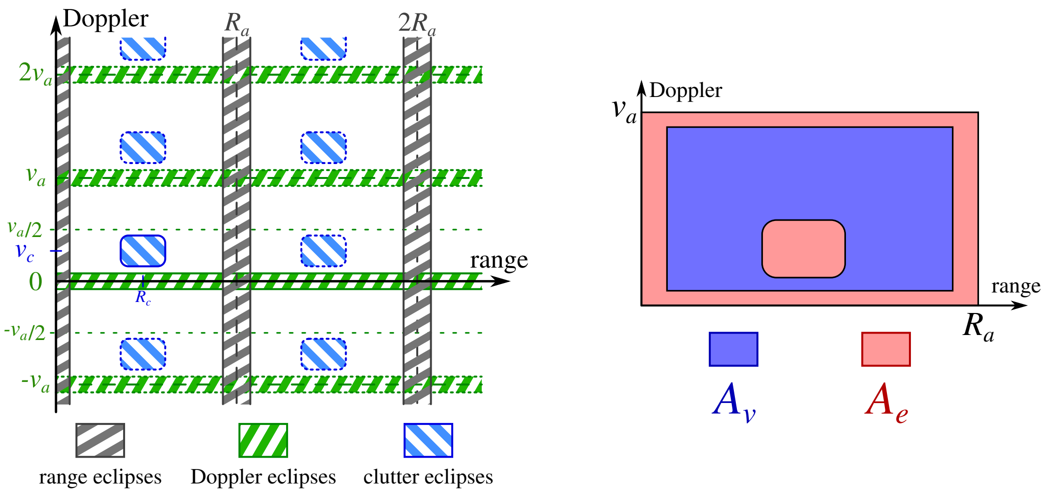

Figure 1.16: range-Doppler map with eclipses for a given burst (left) and its visible and occulted areas (right)

- *Burst integration*: In double threshold detection, a detection is validated if and only if there are at least "K_b out of N_b" detections among the bursts, with K_b a chosen threshold. Considering each burst detection as statistically independent, the waveform detection and false alarm probabilities P_d and P_{fa} are related to the burst detection and false alarm probabilities $P_{d,b}$ and $P_{fa,b}$ by the following relations

$$
\begin{aligned}
P_d &= \sum_{k=K_b}^{N_b} \binom{N_b}{k} {P_{d,b}}^k (1 - P_{d,b})^{N_b-k} \\
P_{fa} &= \sum_{k=K_b}^{N_b} \binom{N_b}{k} {P_{fa,b}}^k (1 - P_{fa,b})^{N_b-k}
\end{aligned}
\tag{1.8}
$$

1.5.3 Radar eclipses and clutter

A radar in operation usually has blind areas, also called *eclipses*, shown in Figure 1.16 :

- *Range eclipses*: Along the distance axis, a mono-static radar cannot receive while emitting. Either the same antenna is used for both emission and reception, or different antennas are used but will interfere with each other. Thus there is a blind interval during each pulse emission, see Figure 1.17. Since a burst is a sequence of pulses, this blind interval

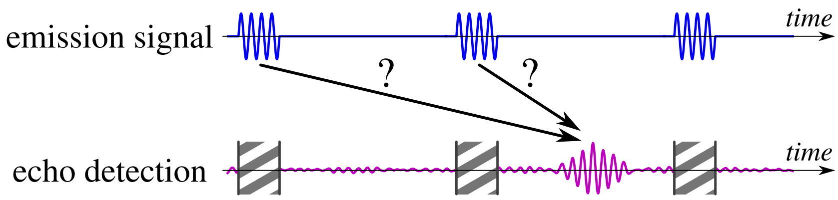

Figure 1.17: Range eclipses and ambiguity on which pulse the echo originates from

is replicated during each pulse emission. Distance eclipse are located at ranges $kR_a, k \in \mathbb{N}$ with $R_a = \frac{c\,T_p}{2}$.

R_a is called the *range ambiguity*: if a target is located at $R > R_a$, further than the ambiguity range, then a reflected pulse is received only after the next pulse has been emitted, leading to an ambiguity on which of the two pulses reflection has actually been received, see Figure 1.17. Range measurements from a burst are only known "modulo R_a".

- *Doppler eclipses*: the target radial speed can be estimated using Doppler filtering. In general, the entire surrounding environment (ground, sea, trees, etc.) also reflects back the radar signal with no (or little) radial speed. The zero speed estimation is polluted by the entire environment. In practice, it is impossible to discriminate a non-moving target of interest from the rest of the environment. Because Doppler filtering is essentially a form of "speed sampling", aliasing occurs for speeds over a certain value v_a, known as the *Doppler ambiguity*, and target faster than v_a appears to be slower (or even moving away). Because of aliasing, the zero-speed blind area is also replicated along the Doppler axis.

- *Clutter eclipses*: environmental elements hindering detection are called *clutter*. The zero-speed Doppler eclipse is usually due to ground or sea clutter, which are immobile. However, certain elements, such as rain, can be moving due to wind, and occult areas on the clutter map which are beyond the zero speed.

The *eclipse coefficient* α is defined as the ratio of all eclipsed areas over the total area of the range-Doppler map

$$\alpha = \frac{A_e}{A_v + A_e}$$

as represented in Figure 1.16.

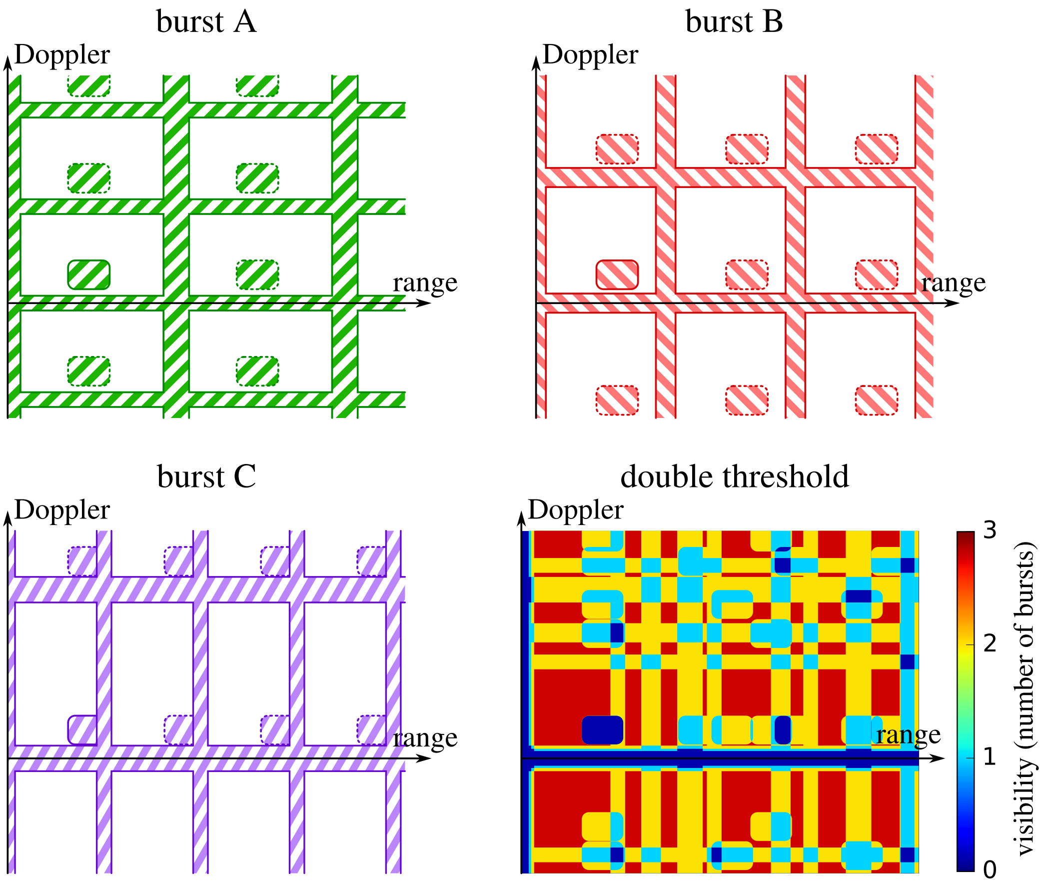

Figure 1.18: Total eclipse on range-Doppler map for different bursts, and (bottom right) waveform double threshold visibility

Positions of Doppler eclipses and distance eclipses beyond the first occurrence can be controlled by changing the pulse repetition interval and the number of pulses inside a burst. Those parameters usually vary from burst to burst, to ensure that for most speed-range positions on the map, a reasonable number of bursts inside the waveform can still detect the target, see Figure 1.18.

In an approximative statistical model, the detection location on the range-Doppler map can be considered uniformly random and independent between bursts, with a probability $(1-\alpha)$ to be visible and a probability α to be occulted in an eclipsed area. So the probability that "n among N_b" bursts are visible is $\binom{N_b}{n}(1-\alpha)^n\alpha^{N_b-n}$. Considering that each burst has the same detection probability $P_{d,b}$, then the probability of having K_b successful detections out of n visible bursts is $\sum_{k=K_b}^{n}\binom{n}{k}{P_{d,b}}^k(1-P_{d,b})^{n-k}$. The waveform detection probability, and by similar reasoning false alarm probability, accounting for eclipse coefficient α are

$$\begin{aligned} P_d &= \sum_{n=N_b}^{N_b} \binom{N_b}{n} (1-\alpha)^n \alpha^{N-n} \sum_{k=K_b}^{n} \binom{n}{k} {P_{d,b}}^k (1-P_{d,b})^{n-k} \\ P_{fa} &= \sum_{n=N_b}^{N_b} \binom{N_b}{n} (1-\alpha)^n \alpha^{N-n} \sum_{k=K_b}^{n} \binom{n}{k} {P_{fa,b}}^k (1-P_{fa,b})^{n-k} \end{aligned} \tag{1.9}$$

This model requires the assumption that burst detections are independent for the same target position on the range-Doppler map, which in practice is unlikely to be accurate, especially for target close to the range-Doppler map origin, i.e. slow targets close to the radar location. However it can be used as a simple method to approximate the energetic impact of clutter.

Within this model, the waveform required signal-to-noise ratio to achieve desired detection and false alarm probabilities also depends on the eclipse coefficient α and is noted $s_w(P_d, P_{fa}, \alpha)$.

1.6 Dwell model and range computation

Radar detection depends on both the radiation pattern defined by the spatial model and the waveform defined by the time model. The electromagnetic signal emitted by each radiating elements is the product of the signal waveform "spatially" modulated by the illumination law.

To achieve detection of a given target, one must modulate the phased array radiating elements with an adequate illumination law, and then feed an adequate waveform signal in the radiating elements. In terms of optimization, the illumination law and the signal waveform can be viewed as "variables", meaning they are the physical values through which radar detection can be controlled. Informally, the illumination law controls "where the radar looks" and the waveform controls "how the radar listens in that direction".

The combination of a given illumination law and a given waveform is called a *dwell*

$$d = (\{a_{k,l}\}, w)$$

Computing the detection range of a given dwell at desired detection and false alarm probabilities P_d and P_{fa} in direction (az, el) can be done using the radar equation with the model described in this chapter. The radar Equation 1.1 can be reformulated to express the detection range in function of the other parameters

$$R^4 = \frac{P_p\ f_w\ T_w\ g_e\ g_r\ {\lambda_w}^2\ \sigma}{(4\pi)^3\ s_w\ L_u\ {L_s}^2} \tag{1.10}$$

which can be further simplified:

- The radar peak emission power P_p, the reception gain of digital beamforming $g_r = g_{DBF}$ and the uniform losses L_u are constants of the system by design and can be computed as a unique term

$$K_r = P_p \; g_r \; (4\pi)^{-3} \; {L_u}^{-1}$$

- The dutycycle f_w, duration T_w, carrier wavelength λ_w are constants[1] of the waveform can be computed as a unique term

$$K_w = f_w \; T_w \; {\lambda_w}^2$$

The simplification reduces the equation to

$$R^4 = K_r \; K_w \; g_e \; \sigma \; {s_w}^{-1} \; {L_s}^{-2} \tag{1.11}$$

The scanning direction cosines coordinates can be expressed from the direction operational coordinates and the radar tilt angle t using Equation 1.4

$$\begin{aligned} u &= \cos(el)\sin(az) \\ v &= \sin(el)\cos(t) - \sin(t)\cos(az)\cos(el) \\ w &= \sin(el)\sin(t) + \cos(t)\cos(az)\cos(el) \end{aligned}$$

which can then be used to compute

- The emission gain from Equation 1.3, knowing the waveform carrier wavelength

$$g_e(u,v) = \sum_{k=0}^{K-1}\sum_{l=0}^{L-1} a_{k,l} \, e^{j\phi_{k,l}} \, e^{j2\pi\frac{ld_x u + kd_y v}{\lambda_w}}$$

- The scanned losses as ${L_s}^{-2} = w^2$

The waveform required signal-to-noise ratio s_w for achieving desired detection and false alarm probabilities can be computed through measurements or simulations. In our experiments, we used the waveform model presented in the previous section for computing this signal-to-noise ratio.

[1] In practice, the carrier wavelength can changes between bursts, due to frequency agility, impacting the antenna gain. Corrections in the illumination law can somehow compensate those changes. The present model makes the simplifying assumption that the carrier does not change

To achieve detection and false alarm probabilities P_d and P_{fa} out of the double threshold detector, each burst detection and false alarm probabilities must be the solutions $P_{d,b} = x$ and $P_{fa,b} = y$ of the system

$$\left\{ \begin{array}{llll} \sum_{n=K_b}^{N_b} \binom{N_b}{n} (1-\alpha)^n \alpha^{N_b-n} & \sum_{k=K_b}^{n} \binom{n}{k} x^k (1-x)^{n-k} & -P_d & = 0 \\ \sum_{n=K_b}^{N_b} \binom{N_b}{n} (1-\alpha)^n \alpha^{N_b-n} & \sum_{k=K_b}^{n} \binom{n}{k} y^k (1-y)^{n-k} & -P_{fa} & = 0 \end{array} \right.$$

with N_b the number of bursts in the waveform, K_b the detections threshold, and α the eclipse ratio. Analytically, the solutions are the roots of high-degree polynomials which in general might not have a closed form. Numerically, the solutions can be find by root-finding line search [5]. The required signal-to-noise ratio by burst s_b can be deduced from the Swerling formulas (1.7), either analytically or by numerical root-finding, and so the waveform required signal-to-noise ratio is

$$s_w = N_b \; s_B$$

Knowing all terms of Equation 1.11, the detection range of dwell d can be computed.

Chapter 2

Optimization theory and computational complexity

2.1 Introduction and literature

Optimization theory is the area of mathematics which focus on efficiently finding solutions of good-quality respectively to a certain metric to a formally defined problem. The topics in optimization theory are broad, encompassing:

- *Modelling*: "how to formalize a real life problem in mathematical terms",
- *Complexity theory*: "how to consistently define the difficulty of a mathematical problem"
- *Algorithmics*: the art of designing systematic procedures to solve mathematical problems.

The first step when applying optimization to a real-life problem is to grasp the problem true nature, its underlying mathematical structure. From this knowledge the problem can be linked to known classical problems in the literature, and knowing the problem properties will lead to practical design of balanced algorithm between efficiency, usability and accuracy.

Optimizing radar scanning can be informally described as the research of an efficient *radar search pattern*, a collection of dwells achieving certain detection requirements. Since multi-functions radar must deal with other tasks in addition to scanning, being able to perform scanning as efficiently as possible is desirable. Quantitatively, an efficient search pattern should achieve detection with minimal radar time-budget. This problem is thus structurally a covering problem, finding a "smallest" subset of elements which covers a space.

In optimization, the most general form of covering problems is known as *set covering*, a classical problem in combinatorial optimization. The objective is to cover a set of elements, called the universe, using a minimum number of available covers. The theoretical problem is known to be generally NP-hard to solve [10], and is often encountered in industrial processes and real-life problems. It has been extensively studied since its description as one of Karp's 21 classical NP-complete problems [11], which is the common class for difficult industrial problems. The set cover problem is also hard to approximate: while the greedy heuristic has a logarithmic approximation ratio in the number of constraints in both weighed and unweighed cases [12, 13], the problem cannot really be more efficiently approximated unless P=NP [14, 15, 16]. Alternate approximation bounds have also been found using randomized rounding algorithms [17].

Due to its theoretical hardness, a part of the research has focused on finding empirically efficient methods, even with exponential worst-case theoretical complexity. Branch-and-bound approaches based integer programming can be rather efficient [18, 19], and most exact methods are variation of the branch-and-bound scheme. Various metaheuristics have also been applied to the problem [20, 21]. Certain covering problems which can be viewed as specific geometric cases and weaker formulations of the set cover problem can have stronger properties, even be solvable or approximated in polynomial time [22, 23, 24, 25].

In the case of radar covering, combinatorial problems modelling bi-dimensional radars have strongly polynomial complexity, meanwhile three-dimensional radars models are NP-hard to optimize, as will be shown in this chapter.

2.1.1 Decision problems and complexity classes

For each optimization problem, there is a corresponding decision problem, which puts the optimization problem into the form of a "yes/no" question. The question is usually, for a given value $K \in \mathbb{Z}$: "is there is solution to the minimization (maximization) problem whose value is smaller (higher) than K ?". Decision problem are one of the fundamental concepts in computational complexity theory and are used to define complexity classes. The most common classes for real-life problems are P and NP, for which informal definitions are given below (see [10, 26] for formal definitions).

P is the class of all decision problems which can be solved in polynomial time on a deterministic computer machine. That is, for any problem, there is an deterministic algorithm which can solve any instance of the problem in polynomial time and answer to the question "is there a solution with better value than a given K ?".

NP stands for non-deterministic polynomial, and is the class of all problems which can be solved in polynomial time on a non-deterministic machine, a machine in which multiple choices can be explored in parallel. A more sensible definition is that for the same problem, a deterministic machine would take polynomial time to check one given solution and answer the question "does this solution has better value than K ?". A non-deterministic machine can use the same algorithm to check all solutions in parallel in the same time. NP is often described as the class of problems for which a solution is easy to check (polynomial time), but hard to find (exponential time) in the current state of the art.

Furthermore, a problem is said to be NP-hard, if any problem in NP can be reduced to said problem through a polynomial reduction. A NP-hard problem is thus at least as hard as the hardest problems in NP (but could be harder, as there are NP-hard problems not in NP). A polynomial algorithm for any NP-hard problem could be used to solve any NP problem in polynomial time. A problem that is both in NP and NP-hard is called NP-complete.

By extension, an optimization problem is said to be in P/NP (sometimes in PO/NPO), if its decision version is in P/NP.

2.2 Problem statement and modelling

2.2.1 Set cover problem

Let $G = \{g_i\}$ be a set of elements, called the *universe set*. Let $\mathcal{C} = \{C_j \subset G\}$ be a collection of subsets in G, a set cover is a sub-collection $\mathcal{S} \subset \mathcal{C}$ whose union covers the universe: $\bigcup_{C \in \mathcal{S}} C = G$.

The decision form of the *set cover problem* asks whether for a given $K \in \mathbb{N}$ there exists a set cover $\mathcal{S} \subset \mathcal{C}$ with cardinality inferior to K, i.e. $|\mathcal{S}| \leq K$. An instance of the set cover problem is described by the system $(G, \mathcal{C}, K)$. The optimization form, called *minimum set cover problem*, consists in finding a minimum-size set cover:

$$\begin{array}{rl} \min & |\mathcal{S}| \\ s.t. & \forall g_i \in G, \exists C \in \mathcal{S},\ g_i \in C \\ & \mathcal{S} \subset \mathcal{C} \end{array} \tag{2.1}$$

If each element $C_j \in \mathcal{C}$ has an associated cost $T_j \in \mathbb{N}^*$, the problem of finding a set cover with minimal aggregate cost $\sum_{C_j \in \mathcal{S}} T_j$ is called the *weighted set cover problem*. The previous cases correspond to $\forall j, T_j = 1$, and are said to be *unweighted*.

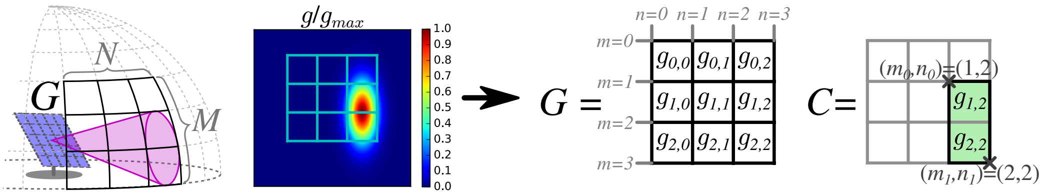

Figure 2.1: Dwell radiation pattern (left), detection grid G and detection discrete cover C (right)

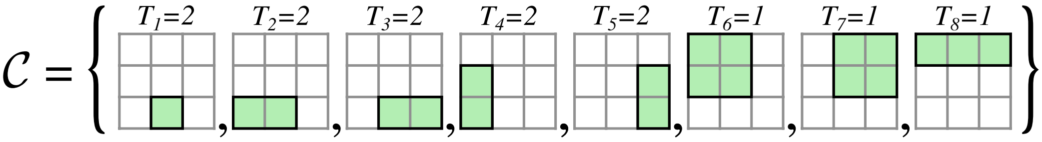

Figure 2.2: Collection $\mathcal{C}$ of available discrete covers in the radar database for a covering problem

From now on we will use a different, radar-based, terminology. The universe set $G = \{g_{m,n}\}$ usually represents a finite bi-dimensional M-by-N regular grid, called the *detection grid*, see Figure 2.1, on which:

- each element $g_{m,n}$ represents a detection cell indexed by $(m, n) \in [0, M[\times[0, N[\subset \mathbb{N}^2$. The grid contains MN detection cells, and corresponds to a certain direction of observation for the radar.

- each node (m, n) represents the intersection of the m-th horizontal line and the n-th vertical line with $(m, n) \in [0, M] \times [0, N] \subset \mathbb{N}^2$. The grid has $(M + 1)(N + 1)$ nodes.

A subset $C \in \mathcal{C}$ represents the detection area of a radar dwell, as presented in Figure 2.1, and is the *(dwell) discrete cover*. The associated cost T_j of a discrete cover C_j is the associated dwell waveform duration. The collection of all available discrete covers forms the *radar dwell cover database*, representing all the discrete covers the radar can emit. A sub-collection of dwell discrete covers, in the radar database, ensuring detection over the entire surveillance space, is called a *radar search pattern*. It corresponds to a set cover of the combinatorial problem. The cost of a radar search pattern is the time required to emit all its dwells in sequential order, and is the aggregate cost of its discrete covers.

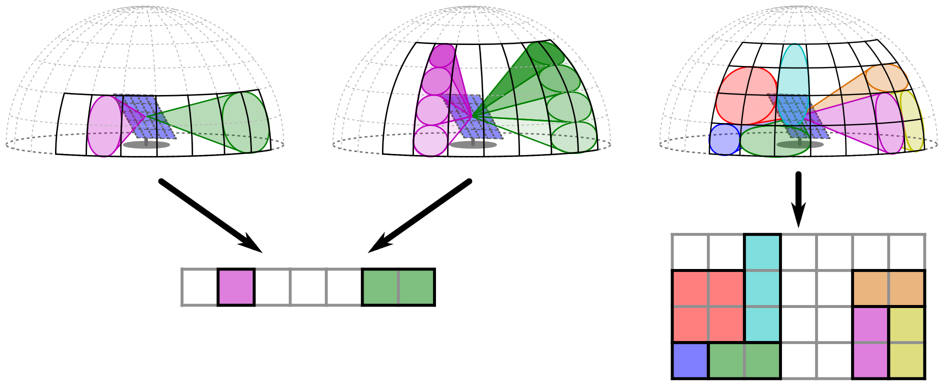

Figure 2.3: Two-dimensional radar (top-left), three-dimensional stacked radar (top-center) and three dimensional radar (top-right), are modelled either as one-dimensional covering problems (bottom-left) or as bi-dimensional covering problem (bottom-right)

2.2.2 Grid dimension

For cover problems in radar applications, the universe set is a grid whose geometry models how the radar scans the environment. Modern antennas can control the direction of emission using beamforming, for which a mathematical model in the case of a bi-dimensional linear phased array antenna was presented in Chapter 1.

Many modern radar systems can perform bi-dimensional beam-steering in azimuth and elevation, such radars are said to be *three-dimensional*, as they work in with three coordinates: azimuth, elevation and range.

There exists radars performing only azimuthal beam-steering, working only with the two dimensions of azimuth and range, either because the radar beam covers the entire elevation at once, or either because the surveillance is very narrow on the horizon. Such radars are said to be *two-dimensional*. There also three-dimensional radars stacking multiple beams in elevation, which can be viewed as two-dimensional radars from a modelling point of view. An example of each of those radar is displayed in Figure 2.3.

Figure 2.3 also presents the two possibilities for modelling the detection grid in radar covering problems:

- in two-dimensional models, the detection grid has only one dimension. This corresponds to a particular case where $M = 1$ and $N \in \mathbb{N}$.
- in three-dimensional models, the detection grid has two dimensions. This is the general case where $(M, N) \in \mathbb{N}^2$.

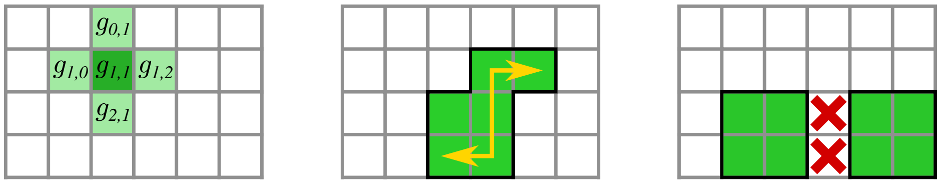

Figure 2.4: Set of neighbours $\{g_{0,1}, g_{2,1}, g_{1,0}, g_{1,2}\}$ for a given cell $g_{1,1}$ (left), connected shape (center) and disconnected shape (right)

2.2.3 Connected and disconnected dwells

Usually, a radiation pattern will form a single beam and have a connected shape, which are simpler to manipulate from an engineering point of view. It also minimize the energy lost in side-lobes.

Single-beam dwells result in connected discrete covers for the associated grid cover problem. The definition of a connected subset on grid G is based on cell neighbourhood: let $g_{a,b} \in G$ and $g_{c,d} \in G$, then $g_{a,b}$ and $g_{c,d}$ are neighbours if and only if:

$$(a = c) \wedge (|b - d| \leq 1) \bigvee (|a - c| \leq 1) \wedge (b = d)$$

In other words, the neighbours to an element $g \in G$ are its four adjacent cells, as presented in Figure 2.4.

A subset on the grid is connected if for any two cells in the subset, there is path between them moving from neighbour to neighbour. A subset which is not connected is said to be disconnected. Both cases are displayed in Figure 2.4.

An interesting case of connected covers are rectangular-shaped covers. In radar engineering, a feasible radiation pattern is synthesized to fill as closely as possible a desired shape. Rectangular shapes are usually good candidates.

On the grid, a rectangular-shaped cover is a subset of elements included in a rectangle, uniquely defined by its upper left corner node (m_0, n_0) and its lower right corner node (m_1, n_1), such that $0 \leq m_0 < m_1 \leq M$ and $0 \leq n_0 < n_1 \leq N$. The set representation of a cover defined by corners (m_0, n_0) and (m_1, n_1) is:

$$C = \{g_{m,n},\ (m, n) \in [m_0, m_1[\times [n_0, n_1[\}$$

Figure 2.2 presents an example, cover C_7, where corners are $(m_0, n_0) = (0, 1)$ and $(m_1, n_1) = (1, 2)$.

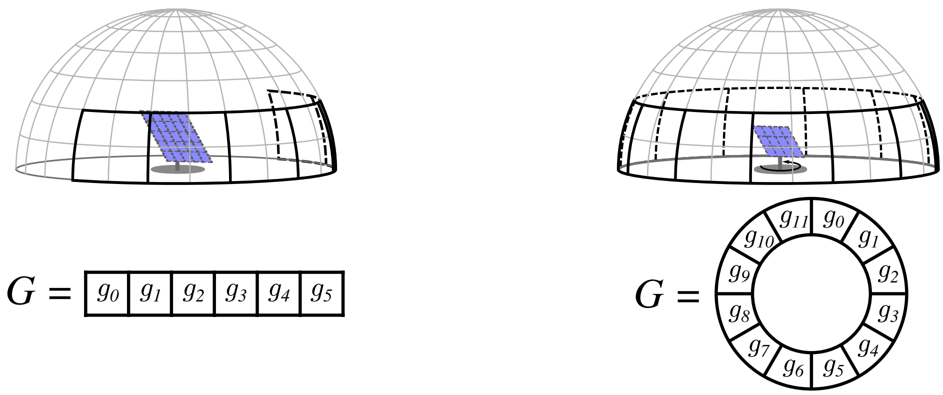

Figure 2.5: Radars with bounded azimuthal range (left) and with full azimuthal range (left)

Rectangles are also more easily synthesized with a bi-linear phased-array antenna, for which the radiation pattern can be separated into an horizontal and a vertical component.

Furthermore, in term of combinatorial complexity, the number of possible rectangles on M-by-N grid

$$\binom{M+1}{2}\binom{N+1}{2} = \frac{MN(M+1)(N+1)}{4} = O(M^2N^2)$$

gives a broad choice of available discrete covers for computing the pattern, while avoiding exponential explosion when increasing the grid resolution.

2.2.4 Azimuthal range and circular grid cover problems

In operational situations, the surveillance space of the radar can be bounded to a certain azimuthal range, for example a fixed panel radar. But the radar can also be required to scan the surrounding space in all directions and have full azimuthal range, using for example a rotating-panel system or multiple fixed-panel system. Limited azimuthal range is modelled by planar grids, while full azimuthal range is modelled by circular grids, as represented in Figure 2.5.

2.3 Integer programming

2.3.1 Matrix formulation

Set cover problems can be written as integer programs by using matrix formulations. Each cover $C \in \mathcal{C}$ can be represented as a binary M-by-N matrix

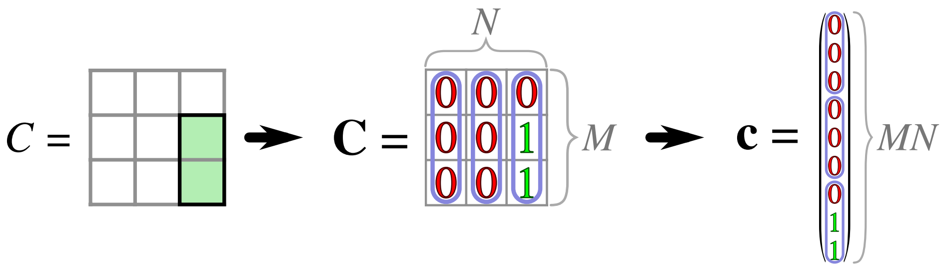

Figure 2.6: Dwell discrete cover (left), its binary matrix representation (center) and its binary vector representation (right)

noted $\mathbf{C}$, or as a binary vector of length MN noted $\mathbf{c}$, as shown in Figure 2.6:

$$\mathbf{C}(m,n) = \mathbf{c}(m+Mn) = \begin{cases} 1 \text{ if } g_{m,n} \in C \\ 0 \text{ otherwise} \end{cases}$$

For each cover $C_i \in \mathcal{C}$, let $x_i \in \{0,1\}$ be the binary selection variable of cover C_i, such that the vector $\mathbf{x} = (x_1, \ldots, x_D) \in \{0,1\}^D$ represents the sub-collection $\mathcal{S} = \{C_i \in \mathcal{C} \ s.t. \ x_i = 1\}$, containing the chosen covers.

Let $\mathbf{T} = (T_1 \cdots T_D)^T$ be the cost vector and let

$$\mathbf{A} = \begin{pmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_D \end{pmatrix} = \begin{pmatrix} \mathbf{C}_1(0,0) & \cdots & \mathbf{C}_D(0,0) \\ \mathbf{C}_1(1,0) & \cdots & \mathbf{C}_D(1,0) \\ \vdots & \ddots & \vdots \\ \mathbf{C}_1(m,n) & \cdots & \mathbf{C}_D(m,n) \\ \vdots & \vdots & \vdots \end{pmatrix}$$

be the cover matrix.

Then the set cover problem can be written as the following integer program:

$$\begin{array}{ll} \min & \mathbf{T}^T.\mathbf{x} \\ s.t. & \mathbf{A} \cdot \mathbf{x} \geq \mathbf{1} \\ & \mathbf{x} \in \{0,1\}^D \end{array} \tag{2.2}$$

where $\mathbf{1}$ is the all-ones vector $(1 \cdots 1)^T$ of length MN. For example, the cover problem represented in Figure 2.2 can described by the following system:

$$\mathbf{A} = \begin{pmatrix} 0&0&0&0&0&1&0&1 \\ 0&0&0&1&0&1&0&0 \\ 0&1&0&1&0&0&0&0 \\ 0&0&0&0&0&1&1&1 \\ 0&0&0&0&0&1&1&0 \\ 1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1 \\ 0&0&0&0&1&0&1&0 \\ 0&0&1&0&1&0&0&0 \end{pmatrix}, \mathbf{T} = \begin{pmatrix} 2&2&2&2&2&1&1&1 \end{pmatrix}^T, \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} \tag{2.3}$$

2.3.2 Linear relaxation

Integer programming is NP-hard to solve [11]. Replacing integer variables by continuous variables transforms the problem into a linear program

$$\begin{array}{ll} \min & \mathbf{T}^T.\mathbf{x} \\ s.t. & \mathbf{A}\cdot\mathbf{x} \geq \mathbf{1} \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{1} \end{array} \tag{2.4}$$

which is called the *linear relaxation* of (2.2). Linear programs can be solved in polynomial time [27]. Any valid solution of the integer program is also a valid solution of its linear relaxation, but the reverse is false. An optimal solution of the linear relaxation is not a valid integer solution in general, and only gives a lower bound for the integer program. Note that the constraint $\mathbf{x} \leq \mathbf{1}$ is in fact unnecessary, since the problem

$$\begin{array}{ll} \min & \mathbf{T}^T.\mathbf{x} \\ s.t. & \mathbf{A}\cdot\mathbf{x} \geq \mathbf{1} \\ & \mathbf{0} \leq \mathbf{x} \end{array} \tag{2.5}$$

has the same optimal solutions as (2.4). Intuitively, for the linear relaxation, a cell is going to be covered by a sum of "fractional" covers (with $x_i < 1$), or as at least one integer cover (with $x_i = 1$) and thus has no need for covers with $x_i > 1$. Any solution with some $x_i > 1$ can be strictly improved by reducing $x_i \leftarrow 1$ while remaining valid and an optimal solution necessarily verifies $\mathbf{x} \leq \mathbf{1}$.

Furthermore, the positivity constraints $\mathbf{0} \leq \mathbf{x}$ can be integrated in the matrix formulation with

$$\mathbf{R} = \begin{pmatrix} \mathbf{A} \\ \mathbf{I} \end{pmatrix} \text{ and } \mathbf{d} = \begin{pmatrix} \mathbf{1} \\ \mathbf{0} \end{pmatrix}$$

by rewriting the linear program as

$$\begin{array}{ll} \min & \mathbf{T}^T.\mathbf{x} \\ s.t. & \mathbf{R}\cdot\mathbf{x} \geq \mathbf{d} \end{array} \tag{2.6}$$

The three formulations of the linear relaxation (2.4), (2.5) and (2.6) are equivalent.

The integer program representing our set cover problem and its linear relaxation have two more interesting properties:

- Easily-checked feasibility: an integer program is feasible if there is at least one solution validating all constraints. It is possible that no valid solution exists if some constraints are conflicting, or if one constraint is impossible. In our case, feasibility is easy to check: the integer program as well as its linear relaxation are feasible if and only if $\mathbf{x}_F = (1 \cdots 1)$ is a feasible solution, i.e. $\mathbf{A} \cdot \mathbf{x}_F = \sum_{i=1}^{D} \mathbf{c}_i \geq \mathbf{1}$:
  - if $\mathbf{x}_F$ is a valid solution, then the problem is feasible by definition.
  - if $\mathbf{x}_F$ is an invalid solution, then there is an invalidated constraint for $\mathbf{x}_F$:
  $$\exists (m,n) \text{ s.t. } \sum_{i=1}^{D} \mathbf{C}_i(m,n) < 1$$
  and since $\forall (i,m,n),\ \mathbf{C}_i(m,n) \in \{0,1\}$:
  $$\exists (m,n) \text{ s.t. } \forall i, \mathbf{C}_i(m,n) = 0$$
  In other words, $\mathbf{A}$ has its $(m + Mn)$-th row filled with zeros, corresponding to a constraint which can be satisfied by no solution. Intuitively, $\mathbf{x}_F$ represents $\mathcal{C}$, the collection of all available covers itself, and if it is an invalid solution, then there is a cell which cannot be covered. This can happen in a real system if there is a cell which cannot be scanned, because of an obstacle or because the radar has not enough power to achieve the desired detection range.
- Boundedness: a recurring question for linear programs is whether they are bounded, that is whether the cost function is bounded (below for minimization) for valid solutions. For the set cover problem, the cost function is positive and thus always bounded below by 0.

2.3.3 Linear programming

There are three important geometrical aspects describing the decision space of the integer and linear programs, shown in Figure 2.7:

- $\mathbf{T}$ is the cost function gradient. The cost function is linear and its gradient is constant. $-\vec{\mathbf{T}}$ is the direction vector of maximum decrease of the cost function.

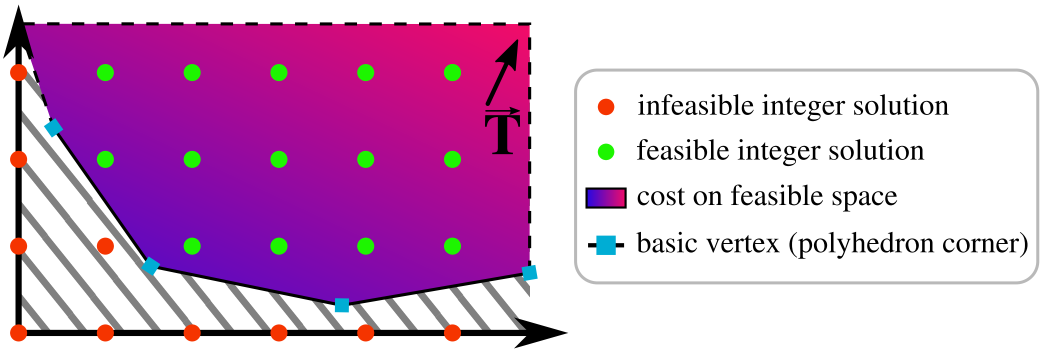

Figure 2.7: Convex polyhedron representing decision space of linear and integer programs (2D example)

- $\mathbf{A}$ is the cover matrix. Each row of $\mathbf{A}$ correspond to a detection constraint on a cell of G. In the decision space, each constraint corresponds to an hyperplane, the limit between the halfspace of solutions validating the constraint and the halfspace of solutions violating the constraint. The intersection of those halfspace forms the convex polyhedron defined by
$$\{\mathbf{x} : \mathbf{A} \cdot \mathbf{x} \geq \mathbf{1}\}$$
- The positivity constraint of the linear relaxation $\mathbf{0} \leq \mathbf{x}$ bounds the values of the valid solutions in the positive orthant $\mathbb{R}_+^D$.

The set of valid solutions for the linear relaxation is the intersection of the valid halfspaces for all constraints, and the orthant $\mathbb{R}_+^D$. Geometrically, it is a convex polyhedron defined by

$$\{\mathbf{x} : (\mathbf{A} \cdot \mathbf{x} \geq \mathbf{1}) \wedge (\mathbf{0} \leq \mathbf{x})\}$$

Each vertex (or "corner") of this polyhedron is a point where at least D hyperfaces of the polyhedron intersect, in other words, a point where D constraints are tight.

Such a point is called a basic solution (or basic vertex) of the linear program. If a linear program is bounded and feasible, then it has a basic optimal solution [27]. Consider a basic optimal solution $\mathbf{x}$ for the reduced linear program in (2.5). This solution has D tight constraints. Let $B \leq MN$ be the number of tight detection constraints. If $B < D$ then there are $Z = D - B$ tight bound constraints, which are of the form $x_i \leq 0$, and thus $x_i = 0$. The corresponding Z variables are called *non-basic* variables and are zeros. The other $D - Z = B$ variables are called *basic* variables and can be

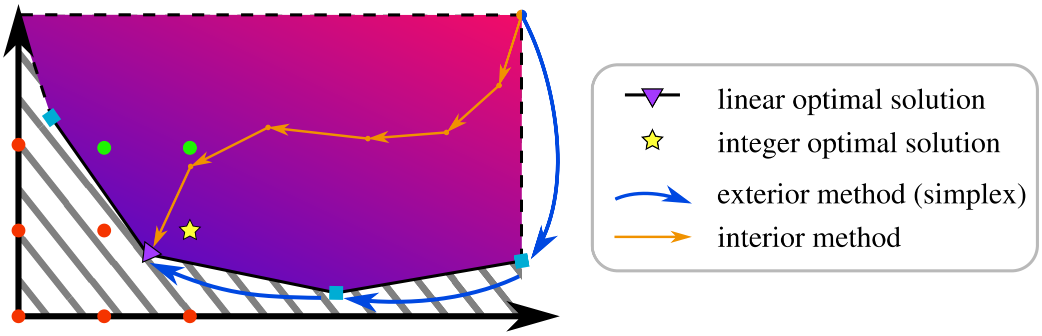

Figure 2.8: Illustration of exterior and interior methods for solving linear programs

non-zero values. Let $\mathbf{x}_B$ be the sub-vector of basic variables. The B tight detection constraints in $\mathbf{A}$ can be written as

$$\mathbf{A}_B \mathbf{x}_B = \mathbf{1} \tag{2.7}$$

where $\mathbf{A}_B$ is the square B-by-B submatrix of $\mathbf{A}$ linking the basic variables $\mathbf{x}_B$ to the tight detection constraints. Furthermore, $\mathbf{A}_B$ is necessarily non-singular: since the hyperplanes of all constraints intersect into a single point, the constraints are linearly independent.

Efficient optimization methods for linear programs generally exploits the feasible polyhedron convexity, and can be viewed as descent methods. Two principal families of algorithms, represented in Figure 2.8, dominate linear programming:

- Exterior descent methods, based on Dantzig's simplex method, which moves from vertex to vertex on the feasible polyhedron until it reaches a basic optimal solution, i.e. a vertex with no decreasing neighbor.

- Interior descent methods, based on Karmarkar's algorithm, which follow a central path. This path is defined by a variable weighing of the cost function and constraint functions, trying to improves the solution values while remaining away from the constraint barriers.

However, descent methods generally cannot be used to solve integer programs, which are not convex since valid integer solutions are isolated points.

2.4 Polynomial-solvable grid cover problems

The general set cover problem and its integer program formulation are both NP-hard to solve. However not all grid cover problems are NP-hard. Certain specific cases of grid cover problem can be solved in polynomial-time. All such problems are one-dimensional ($M = 1$). Interestingly, greedy method or linear programming can solve to optimality certain but not all cases, despite the fact that all problems presented in this section can be solved by a strongly polynomial algorithm based on dynamic programming.

2.4.1 Line cover problem

Consider a two-dimensional radar model, with bounded azimuthal range, using only connected radiation patterns. This model corresponds to two-dimensional fixed radars or three-dimensional fixed radars using stacked-beam. In the associated combinatorial problem, the detection grid has only one dimension and all discrete covers are connected sets. A one-dimension grid can be viewed as a line segment, and since the discrete cover are connected sets in this line segment, they represent intervals (see Fig. 2.9). In this case, finding an optimal radar search pattern can be called a *line cover* problem.

Greedy method

For unweighed line covering, where all covers have the same cost $T_j = 1$, a straightforward algorithm to solve this problem is the greedy method: among intervals covering the first not-yet-covered detection cell, choose an interval covering the furthest cell, and iterate until the line is covered. This method is detailed in Algorithm 1.

The worst case complexity of Algorithm 1 is $O(|\mathcal{C}|^2)$. It can be improved to perform in $O(|\mathcal{C}|\log(|\mathcal{C}|))$ by sorting in advance the available discrete covers in increasing order of their starting point, in which the "while" and "for" loops can be combined in a single loop.

The solution returned by the greedy method is optimal: consider an optimal solution pattern $\mathcal{S}$ of the problem, and $C_a \in \mathcal{S}$ the discrete cover over the first cell, replace C_a by the largest cover C_b which includes the first cell, and pattern $(\mathcal{S} \setminus \{C_a\}) \bigcup \{C_b\}$ still yields an optimal solution. Iterating the process on the rest of the original optimal solution turns it into the greedy

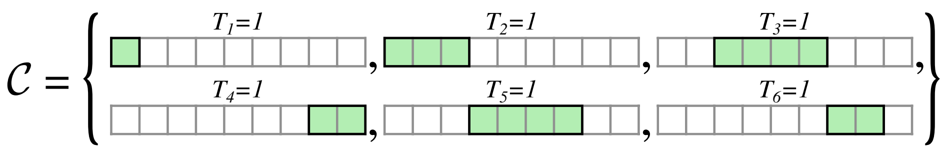

Figure 2.9: Available covers for an example of line cover problem

Algorithm 1 Greedy method

$n \leftarrow 0$ ▷ index of first not-covered cell
$\mathcal{S} \leftarrow \emptyset$ ▷ start with empty pattern

while $n < N$ **do** ▷ loop as long as not all cells are covered
  $l_{best} \leftarrow n - 1$ ▷ index of last covered cell
  for $C_j \in \{C \in \mathcal{C} : g_n \in C\}$ **do** ▷ loop on all covers containing the next cell
    $l \leftarrow$ index of last cell in C_j
    if $l_{best} \leq l$ **then** ▷ Keep the cover of the furthest cell
      $C \leftarrow C_j$
      $l_{best} \leftarrow l$
    end if
  end for
  $\mathcal{S} \leftarrow \mathcal{S} \bigcup \{C\}$ ▷ add cover to solution
  $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C\}$ ▷ remove cover from candidates
  $n \leftarrow l_{best} + 1$ ▷ compute next cell to cover
end while

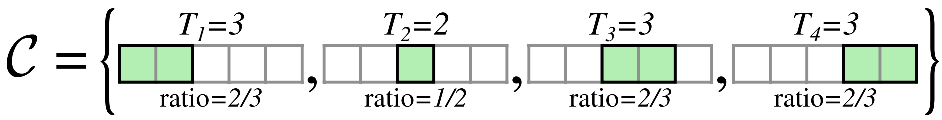

Figure 2.10: Example for sub-optimality of the greedy method in the weighted case

method solution while keeping the same cost. Thus the greedy method returns an optimal solution.

However, the greedy method is sub-optimal for weighted problems. In that case, the logic of the greedy method would be to add at each iteration the cover maximizing the improvement/cost ratio, i.e. the number of newly covered cells over the discrete cover cost. In the unweighed case, a larger discrete cover is strictly better than smaller one, since the former can replace the latter while preserving optimality of the solution. This is no longer true with weighed costs, where a discrete cover with a better improvement/cost ratio (best local choice) can results in sub-optimal solution because of the general structure of the problem (bad global choice), see Figure 2.10 where C_3 has better ratio than C_2 whereas the latter must be used to construct an optimal solution. The greedy method returns the solution $\{C_1, C_3, C_4\}$ with cost 9, whereas the optimal solution $\{C_1, C_2, C_4\}$ has cost 8.

Dynamic programming

The reason why the greedy method is optimal in the unweighed case is because the problem possessed an *optimal substructure*, which means that an optimal solution can be constructed by combining solutions of substructures in the original problem. This type of structure is generally exploited in *dynamic programming*. This approach is particularly efficient if the problem substructure can be broken down into a polynomial number of sub-problems.

Dynamic programming generalizes the iterative approach of the greedy method, and unlike the latter, returns an optimal solution even for weighed line covering. In dynamic programming, an optimal pattern covering for the first n cells is built from an optimal pattern covering the first $k\,(< n)$ cells. The n-th sub-problem is defined as "covering $\{g_i : 0 \leq i < n\}$, i.e. the first n cells". Iterating the process on n yields a valid solution. The method is detailed in Algorithm 2.

Algorithm 2 Dynamic programming for line cover

```
S_0 ← ∅                                         ▷ the solution for covering no cells is the empty set

for n ∈ {1, ..., N} do                          ▷ loop on all sub-problems
    T_best ← +∞
    for C ∈ {C ∈ 𝒞 : g_{n−1} ∈ C} do            ▷ loop on all covers containing next cell
        k ← index of first cell in C
        S ← S_k ⋃{C}                            ▷ construct candidate solution
        T_S ← Σ_{C_j ∈ S} T_j                   ▷ compute candidate cost
        if (T_S ≤ T_best) then
            S_n ← S                             ▷ keep best valid solution for the n-th sub-problem
            T_best ← T_S
        end if
    end for
end for
```

The algorithm requires $O(N|\mathcal{C}|)$ computational steps. The returned solution pattern is optimal: consider an optimal pattern $\mathcal{S}_n$ for the n-th sub-problem, then $\mathcal{S}_n$ contains a discrete cover C including cell g_{n-1} which starts at some cell g_k. Thus $\mathcal{S}_n \setminus \{C\}$ is a valid pattern for the k-th sub-problem. Let $\mathcal{S}_k$ be an optimal pattern for the k-th sub-problem. Then $\mathcal{S}_k \cup \{C\}$ is a valid pattern for the n-th sub-problem. By optimality of $\mathcal{S}_n$

$$\sum_{C_j \in \mathcal{S}_n} T_j \leq \sum_{C_j \in \mathcal{S}_k \cup \{C\}} T_j$$

by optimality of $\mathcal{S}_k$

$$\sum_{C_j \in \mathcal{S}_n \setminus \{C\}} T_j \geq \sum_{C_j \in \mathcal{S}_k} T_j$$

and by combining the two equations

$$\sum_{C_j \in \mathcal{S}_n \setminus \{C\}} T_j = \sum_{C_j \in \mathcal{S}_k} T_j$$

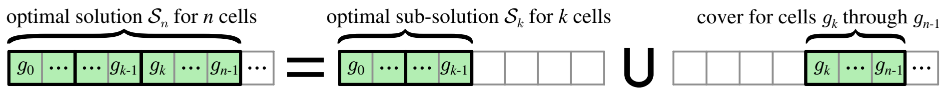

Figure 2.11: Line covering optimal substructure of the n-th sub-problem

so $\mathcal{S}_n \setminus \{C\}$ is an optimal pattern for the k-th sub-problem and $\mathcal{S}_k \cup \{C\}$ is an optimal pattern for the n-th sub-problem.

Any optimal pattern for a given sub-problem is the union of an optimal pattern for a smaller sub-problem and a cover, as shown in Figure 2.11. By testing each combination of a cover and its complementary optimal sub-pattern, dynamic programming sequentially solves all the sub-problems to optimality.

Unlike the greedy method, the complexity of dynamic programming depends on the grid size N. This will be discussed in more details in 2.4.3.

Linear program integrality

Another approach for solving line cover problems is based on the integer program linear relaxation. There are some cases when linear programming can be used to solve exactly integer programs.

The matrix $\mathbf{A}$ is called unimodular if it is invertible and $\det \mathbf{A} \in \{-1, 1\}$. A direct consequence of Laplace's formula $\mathbf{A}^{-1} = (\det \mathbf{A})^{-1} \operatorname{com} \mathbf{A}^T$, with $\operatorname{com} \mathbf{A}$ being the cofactor matrix of $\mathbf{A}$, is that both $\mathbf{A}$ and its inverse $\mathbf{A}^{-1}$ have integer coefficients.

The matrix $\mathbf{A}$ is said to be totally unimodular if any square regular sub-matrix $\mathbf{A}_B$ in $\mathbf{A}$ is unimodular. So any basic solution $\mathbf{x}_B = {\mathbf{A}_B}^{-1} \cdot \mathbf{1}$ of Equation (2.7) has integral values. In such cases, all the vertices of the convex polyhedron represented in Figure 2.7 are integral points, and a basic optimal solution of the linear program is also a valid optimal solution of the integer program. Integer programming is reduced to linear programming, which has polynomial complexity, as finding a basic optimal solution to a linear program can be done in polynomial time [28].

And fortunately, in the case of line covering, the constraint matrix $\mathbf{A}$ has the *consecutive-ones* property, i.e. in a column of $\mathbf{A}$, all values are zeros or ones, and more importantly all the ones are consecutive. This type of matrices are also called *interval matrices* and are known to be totally unimodular [29]. Using linear programming, line covering can be solved in polynomial time.

2.4.2 Circle cover problem

In the case of two-dimensional radar model with full azimuthal range, the detection grid is no longer bounded and can be represented as a circle, with no beginning and no end, as displayed in Figure 2.5. Dynamic programming can still be used to achieve polynomial-time resolution.

The problem still has an optimal substructure. Consider the detection cells as being numbered in clockwise order starting from any cell: $G = \{g_0, \dots, g_{N-1}\}$ with cell g_{N-1} and g_0 being neighbours. The (n, w)-th sub-problem is defined as "covering $\{g_k : k = n + i \mod N, 0 \leq i < w\}$, i.e the w cells in clockwise order starting with g_n". A sub-problem can be described by its (starting) index $n \in \{0, \dots, N-1\}$ and its width $w \in \{1, \dots, N\}$. The substructure of circle covering can be viewed as splitting the problem into all possibles segments on the circle.

Algorithm 3 Dynamic programming for circle cover

```
for n ∈ {0, ..., N − 1} do
    S_{n,0} ← ∅                         ▷ the solution for covering no cells is the empty set
end for

for n ∈ {0, ..., N − 1} do                              ▷ loop on all sub-problems
    for w ∈ {1, ..., N} do
        T_best ← +∞
        l ← n + w − 1  mod N                  ▷ compute index of the next cell to cover
        for C ∈ {C ∈ 𝒞 : g_l ∈ C} do         ▷ loop on all covers containing next cell
            k ← index of clockwise left-most cell in C

            if k − n  mod N ≤ l − n  mod N then       ▷ check if "n ≤ k ≤ l" clockwise
                s ← k − n  mod N                    ▷ complementary sub-pattern width
                S ← S_{n,s} ⋃ {C}                     ▷ construct candidate solution
            else                                    ▷ otherwise "k < n ≤ l" clockwise
                S ← {C}                           ▷ C suffices to solve current problem
            end if

            T_S ← ∑_{C_j ∈ S} T_j                          ▷ compute candidate cost
            if (T_S ≤ T_best) then
                S_{n,w} ← S              ▷ keep best valid solution for (n, w)-th sub-problem
                T_best ← T_S
            end if
        end for
    end for
end for
```

Algorithm 3 requires $O(N^2|\mathcal{C}|)$ computational steps and returns an optimal solution pattern: consider an optimal pattern $\mathcal{S}_{n,w}$ for the (n, w)-th sub-problem with $w \geq 1$, then $\mathcal{S}_{n,w}$ contains a discrete cover C starting at cell g_k (in clockwise order) and including cell g_l with $l = n + w - 1 \mod N$. There are two possible situations:

- "$k < n \leq l$" clockwise:
  $\{C\}$ covers the cells $\{g_n, \ldots, g_l\}$ and is an optimal solution of the (n, w)-th sub-problem, so $\mathcal{S}_{n,w} = \{C\}$.

- "$n \leq k \leq l$" clockwise:
  Let $s = k - n \mod N$, then $\mathcal{S}_{n,w} \setminus \{C\}$ is a valid pattern for the (n, s)-th sub-problem. Let $\mathcal{S}_{n,s}$ be an optimal pattern for the (n, s)-th sub-problem. Then $\mathcal{S}_{n,s} \cup \{C\}$ is a valid pattern for the (n, w)-th sub-problem. By optimality of $\mathcal{S}_{n,w}$

  $$\sum_{C_j \in \mathcal{S}_{n,w}} T_j \leq \sum_{C_j \in \mathcal{S}_{n,s} \cup \{C\}} T_j$$

  by optimality of $\mathcal{S}_{n,s}$

  $$\sum_{C_j \in \mathcal{S}_{n,w} \setminus \{C\}} T_j \geq \sum_{C_j \in \mathcal{S}_{n,s}} T_j$$

  and by combining the two equations

  $$\sum_{C_j \in \mathcal{S}_{n,w} \setminus \{C\}} T_j = \sum_{C_j \in \mathcal{S}_{n,s}} T_j$$

  so $\mathcal{S}_{n,w} \setminus \{C\}$ is an optimal pattern for the (n, s)-th sub-problem.

Any optimal pattern for a given sub-problem is either a unique cover, or the union of an optimal pattern for a smaller sub-problem and a cover, see Figure 2.12.

Informally, Algorithm 3 can be viewed as applying N times Algorithm 2, each time taking a different cell as the starting cell of the "line to cover". Another approach could be to start with an initial solution $\mathcal{S} = \{C\}$ and apply Algorithm 2 for covering the rest of the circle $G \setminus C$, which can be laid out as a line; doing this for each available cover $C \in \mathcal{C}$ and taking the best solution overall will produce an optimal solution. This approach would require $O(N|\mathcal{C}|^2)$. An improved algorithm is presented at the end of the section.

Integrality gap

Linear programming, however, cannot be used to solve circle covering. In this problem, the matrix constraint $\mathbf{A}$ encoding the discrete covers can be non-unimodular. The simplest problem instance for which this situation appears is displayed in Figure 2.13.

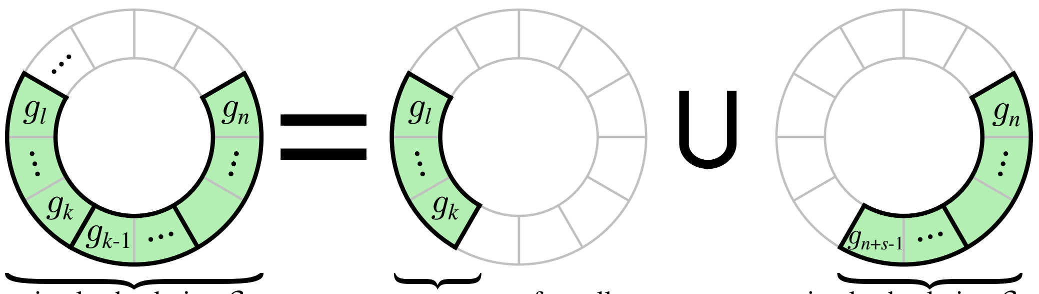

Figure 2.12: Circle covering optimal substructure of the (n, w)-th subproblem

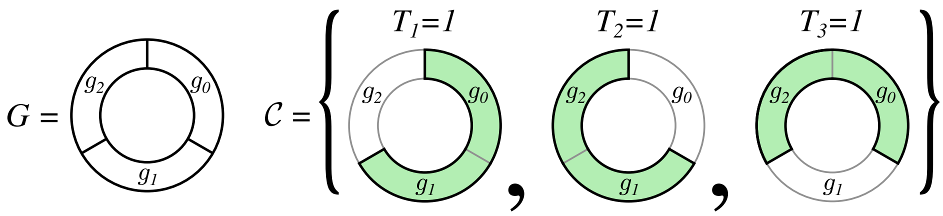

Figure 2.13: Example of non-integral circle cover problem

The relaxed linear program has constraint matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

with $\det(\mathbf{A}) = 2$ and yields the unique fractional optimal solution $\mathbf{x}_L = (\frac{1}{2}\ \frac{1}{2}\ \frac{1}{2})^T$, which combines a weighing of all three covers to produce the optimal fractional pattern, and is strictly better than an integral optimal solution, say $\mathbf{x}_I = (1\ 1\ 0)^T$. The difference of cost between both solutions is called the *integrality gap*, here $\mathbf{T}^T \cdot (\mathbf{x}_I - \mathbf{x}_L) = \frac{1}{2}$.

$\mathbf{x}_L$ is the optimal solution to the corresponding *fractional set cover* problem, where pattern solutions can contains fractions of discrete covers. This situation is not dependant on the integer program encoding (i.e. how the problem is transformed into matrix formulation). Problems with a non-null integrality gap are thus non-integral, and are intrinsically unsolvable by linear relaxation.

Interestingly, despite being non-integral, the circle cover problem can be solved in polynomial time through dynamic programming. This gives a practical case of a non-integral problem which is still polynomially solvable.

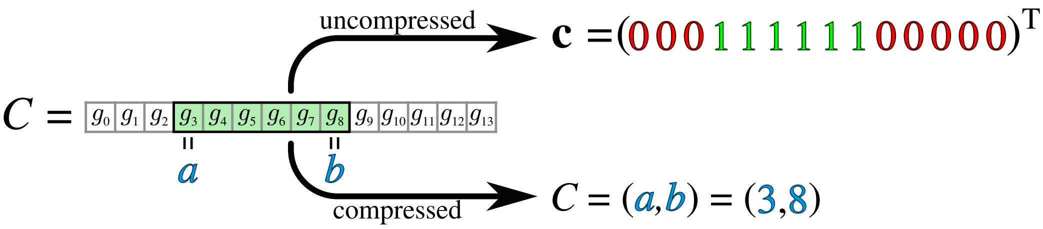

Figure 2.14: Uncompressed boolean vector (top) and compressed logarithmic encoding (bottom)

2.4.3 Logarithmic encoding

All problems presented in this section can be solved in polynomial time using dynamic programming. However, the computational complexity of the corresponding algorithms is polynomial in N, the "grid size". If the problem input is given in matrix formulation, i.e. $\mathbf{c}$ and $\mathbf{A}$, then the encoding size of the input is $|\mathcal{C}|N$, and the algorithm is truly polynomial.

But with interval covers, this encoding scheme is obviously suboptimal, since an interval can be described using only two integers, its starting index a and its ending index b, see Figure 2.14. The number of bits required to encode indexes in $\{0, \ldots, N-1\}$ is $p = \lceil \log_2(N) \rceil$, and the encoding size of a compressed input is $|\mathcal{C}|2p$. For this input size, Algorithm 2 complexity is $O(|\mathcal{C}|2^p)$ and Algorithm 3 complexity is $O(|\mathcal{C}|4^p)$, thus polynomial in the number of input values, but exponential in the number of bits used to encode the values. Such algorithms are said to be *pseudo-polynomial*.

Problems with pseudo-polynomial algorithm can be NP-complete when considering the logarithmic cost, i.e. the computational cost on a deterministic machine using bits to encode values. Such problems are said to be *weakly NP-complete*. An example of a weakly NP-complete problem is the knapsack problem, which also possesses a dynamic programming pseudo-polynomial algorithm [10].

2.4.4 Input reduction

For the cover problems presented in this section, however, it is possible to design a true polynomial algorithm, by using a reduction method before dynamic programming. This method, called *input reduction* is detailed in Algorithm 4 and graphically represented in Figure 2.15.

Input reduction modifies the problem instance by keeping only cells which corresponds to a cover starting index or a post-ending index (cover ending index "plus one"), represented in blue in Figure 2.15. In other words, it only

Algorithm 4 Input reduction

Input : $\mathcal{C} = \{(a_j, b_j)\}_{1\leq j\leq D}$ with a_j, b_j integers encoded with p bits

$G' \leftarrow \bigcup_{(a,b)\in\mathcal{C}}\{a, b+1\}\}$
Sort G' and remove duplicates

$\mathcal{C}' \leftarrow \emptyset$
for $(a, b) \in \mathcal{C}$ **do**
  $a' \leftarrow$ index of a in G'
  $b' \leftarrow$ (index of $b + 1$ in G') $- 1$
  $\mathcal{C}' \leftarrow \mathcal{C}' \cup (a', b')$
end for

Output : $\mathcal{C}' = \{(f'_j, l'_j)\}_{1\leq j\leq D}$ with f'_j, l'_j integers encoded with p bits

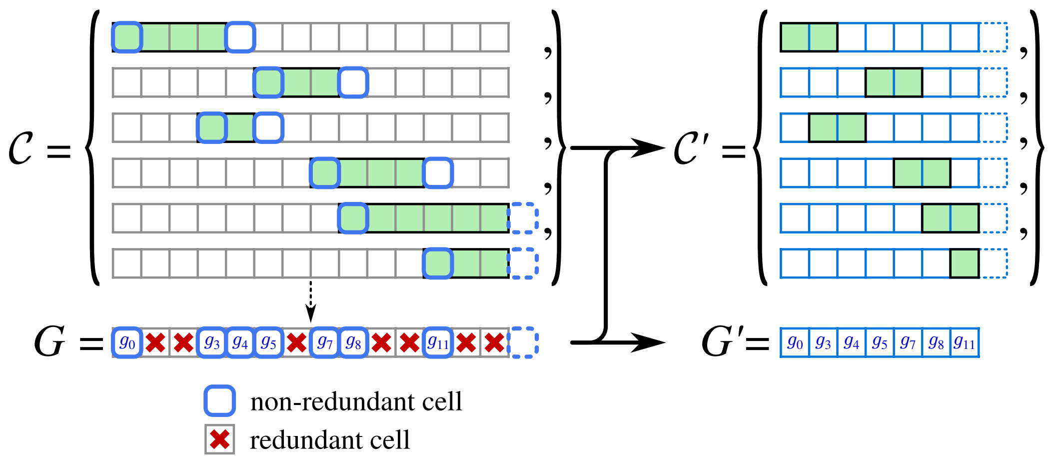

Figure 2.15: Input reduction of the original problem instance (top-left) into a reduced instance (right) by removing redundant detection cells, from the grid (bottom-left)

keeps a cell if it has different candidate covers than the previous cell.

Let g_n be a detection cell which is neither the start nor the post-end cell of a cover. Since n is not a cover starting index, any cover over g_n also covers g_{n-1}. Since n is not the post-ending index of a cover, then $n-1$ is not the ending index of a cover, and any cover over g_{n-1} also covers g_n. So g_{n-1} and g_n have the same covers, i.e. $\forall C \in \mathcal{C}, g_{n-1} \in C \Leftrightarrow g_n \in C$, and the detection constraint over g_n is redundant to the detection constraint over g_{n-1}. Removing the former does not change the problem instance structure, nor the optimal solutions. By induction, it is possible to remove all such detection cells, without changing the instance structure, such that $(G, \mathcal{C})$ and $(G', \mathcal{C}')$ have the same solutions with the same costs.

This approach can be viewed as a form of constraint reduction, as it removes redundant constraints. A redundant constraint corresponds to a detection cell in which "there is no change" relatively to the previous cell, represented in red in Figure 2.15. The constraint reduction implemented in Algorithm 4 is specific to one-dimensional problems. A more general constraint reduction method, suitable for some bi-dimensional problems, is presented in 4.2.2.

Computing and sorting G' takes $O\left(|\mathcal{C}| \log |\mathcal{C}|\right)$ steps or $O\left(p|\mathcal{C}| \log |\mathcal{C}|\right)$ bit operations, with the new grid G' containing at most $2|\mathcal{C}|$ elements. For each cover, finding the new start and end indexes by dichotomic search takes $O\left(\log |\mathcal{C}|\right)$ steps or $O\left(p \log |\mathcal{C}|\right)$ bit operations. Thus the logarithmic cost of input reduction is $O\left(p|\mathcal{C}| \log |\mathcal{C}|\right)$. Each new index can be encoded using

$$\lceil \log_2(2|\mathcal{C}|) \rceil = 1 + \lceil \log_2 |\mathcal{C}| \rceil$$

bits. The output $\mathcal{C}'$ is encoded using at most $2|\mathcal{C}|\left(1 + \lceil \log_2 |\mathcal{C}| \rceil\right)$ bits.

Applying Algorithm 3 on the reduced input takes $O(|\mathcal{C}|^2|\mathcal{C}|) = O(|\mathcal{C}|^3)$ steps. Taking into account the values encoding in $1 + \lceil \log_2 |\mathcal{C}| \rceil$ bits, each step has logarithmic cost $O(\log |\mathcal{C}|)$, so the logarithmic cost of Algorithm 3 is $O\left(|\mathcal{C}|^3 \log |\mathcal{C}|\right)$. By similar reasoning, the logarithmic cost of Algorithm 2 is $O\left(|\mathcal{C}|^2 \log |\mathcal{C}|\right)$.

The overall logarithmic cost of input reduction followed by circle dynamic programming is

$$O\left((|\mathcal{C}|^2 + p)|\mathcal{C}| \log |\mathcal{C}|\right)$$

and by similar reasoning, the overall logarithmic cost for input reduction followed by line dynamic programming is

$$O\left((|\mathcal{C}| + p)|\mathcal{C}| \log |\mathcal{C}|\right)$$

and both line covering and circle covering can be solved in true polynomial time.

Strongly polynomial algorithms

They can in fact be solved in strongly polynomial time, a stronger property. An algorithm is said to be *strongly polynomial* if its arithmetic cost, i.e. the cost when considering arithmetic operations as single computational steps regardless of encoding size, is polynomial in the number of input values. This usually indicates than the number of steps in the algorithm does not depend on the input size and the algorithm performances do not deteriorate too much when inputting large values. A counter-example is Euclid's algorithm for computing the greatest common divisor, whose input is only two numbers, but whose performances decreases when their values grow.

The arithmetic cost of input reduction is $O\left(|\mathcal{C}| \log |\mathcal{C}|\right)$, obtained by replacing $O(p)$ by $O(1)$ in the logarithmic cost, since the p factor only appear as the logarithmic cost of comparisons, and the arithmetic cost of Algorithm 2 is $O\left(|\mathcal{C}|^2\right)$, making the overall cost of input reduction followed by circle dynamic programming

$$O\left(|\mathcal{C}|(1 + \log |\mathcal{C}|)\right)$$

and similarly the arithmetic cost of Algorithm 3 is $O\left(|\mathcal{C}|^3\right)$, making the overall cost of input reduction followed by circle dynamic programming

$$O\left(|\mathcal{C}|^2(1 + \log |\mathcal{C}|)\right)$$

proving that all radar cover problems presented in this section are solvable in strongly polynomial time.

2.5 NP-hard grid cover problems

In this section are presented the remaining grid cover problems, which are all NP-complete in their decision form. This includes problems on a bi-dimensional grid, and problems using disconnected discrete covers, even on a one-dimensional grid. This means that three-dimensional radar models produce NP-hard optimization problems.

2.5.1 Rectangular grid cover problem

In practice, it is reasonable to approximate the grid cover problem associated to a three-dimensional radar by considering only rectangular-shaped covers [30]. The corresponding radiation patterns are simpler to synthesize, and the number of rectangles on the grid grows in $O(M^2N^2)$ keeping the available discrete covers database size reasonable while offering enough choice for producing good quality radar patterns. The corresponding combinatorial

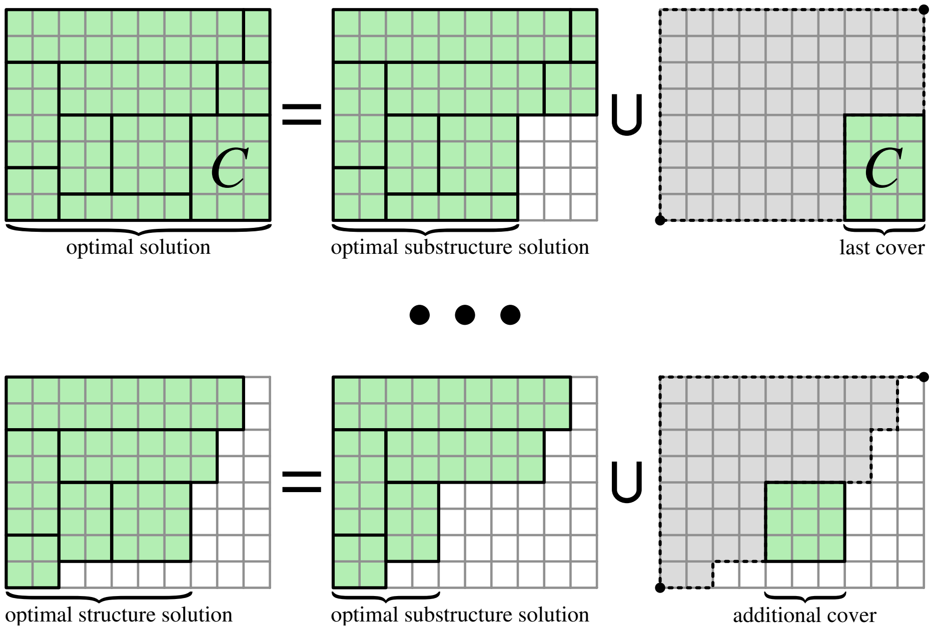

Figure 2.16: Optimal solution decomposition (top), and substructure after multiple decompositions (bottom)

problem amounts to *rectangular grid covering*, and an example instance is presented in Figure 2.2.

Dynamic programming approach

Considering the algorithms presented in the previous section 2.4, a natural attempt to solve rectangular grid covering would be to generalize the dynamic programming approach used on one-dimensional grids to bi-dimensional grids.

Consider an optimal solution for the rectangular grid cover problem. It can be viewed as a combination of a rectangular cover C including the last bottom-right cell and an optimal sub-pattern covering the remaining "top-left" cells. By iterating the decomposition process, the grid cover sub-problem is the cover problem of a "top-left" part of the grid G, see Figure 2.16.

The number of sub-problems is equal to the number of way of cutting G into two sets: a top-left part and a bottom-right part. Equivalently, this is equal to the number of paths between the top-right corner cell and the bottom-left corner cell of G, see 2.17.

A cut is constituted by $N + M$ edges on the grid, with M vertical edges and N horizontal edges. Any cut can be defined uniquely by choosing the N vertical edges (or equivalently M horizontal edges) among the $N + M$

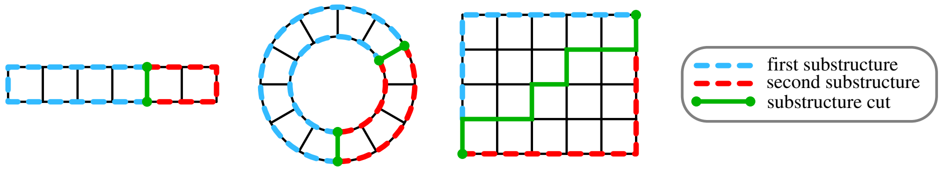

Figure 2.17: Substructure decomposition of line cover problem (left), circle cover problem (center) and grid cover problem (right)

edges. So the number of possible paths between two opposite corners of G, and thus the number of cover sub-problems on G is $\binom{N+M}{N} = \binom{N+M}{M}$. This term grows much faster than for line covering, which has N possible cuts, or circle covering, which has $\binom{N}{2}$ possible cuts, see Figure 2.16.

Let $K = \min\{N, M\}$, then the number of possible cuts can be bound below by the following approximation using Stirling's formula

$$\binom{N+M}{N} \geq \binom{2K}{K} \simeq \frac{\sqrt{2\pi 2K}(2K)^{2K}}{e^{2K}} \left(\frac{e^K}{\sqrt{2\pi K}K^K}\right)^2 = \frac{2^{2K}}{\sqrt{\pi K}}$$

Thus, the number of sub-problems to solve grows exponentially with the grid size: an increase by 10 of the grid size increases the number of sub-problems by approximately $2^{2\cdot 10} \approx 10^6$. Even for small values, the number of sub-problems explodes:

$N = M$	10	20	30	40	50
$\binom{2N}{N}$	$\simeq 10^5$	$\simeq 10^{11}$	$\simeq 10^{17}$	$\simeq 10^{23}$	$\simeq 10^{29}$

Table 2.1: Number of sub-problems

So while theoretically usable for rectangular grid covering, dynamic programming has an exponential complexity for this problem, making the approach rather inefficient. This hints that bi-dimensional grid cover problems are computationally harder than than previous one-dimensional problems.

Note that straightforward linear programming fares no better, as the matrix formulation of a rectangular grid cover problem can also yield a non-totally unimodular matrix $\mathbf{A}$, such as the example given in Figure 2.2, for which the optimal cost is $\frac{11}{2}$, one optimal solution being $\mathbf{x}_L = (0\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2})^T$.

NP-hardness

All problems presented in this chapter can be reduced to the general set cover problem (as formulated in Equation 2.1), or can be reduced to integer programming (as formulated in Equation 2.2), and are thus in NP. However, not all of them are necessary NP-hard, with polynomial counter-examples given in the previous section 2.4.

The classical way for proving NP-hardness is to find a polynomial reduction from an already known NP-hard decision problem to the considered problem. In other words, by showing that there is a polynomial algorithm to turn any instance of the former problem into an instance of the latter. This is called a Karp reduction [26], and by transitivity proves that any problem in NP can be polynomially reduced to the studied problem.

A common candidate for NP-hardness proofs is the vertex cover problem, which is known to be NP-complete [11] and is defined as follow: let $(\mathcal{V}, \mathcal{E})$ be a graph, let $K \in \mathbb{N}$. Is there a subset $\mathcal{U} \subset \mathcal{V}$ with cardinal $|\mathcal{U}| \leq K$ such that $\forall v \in \mathcal{V}, \exists v' \in \mathcal{U}$ with $(v, v') \in \mathcal{E}$? In other words, is there a subset of less than K selected vertices, such that any vertex in the graph has a common edge with a selected vertex ? An instance of the vertex cover problem is defined by the system $(\mathcal{V}, \mathcal{E}, K)$.

The decision version of the rectangular grid cover problem is defined as: let G be a M-by-N rectangular grid, let $\mathcal{C} = \{C_1, \ldots, C_D\}$ be a collection of rectangular-shaped discrete covers on G, and let $F \in \mathbb{N}$. Is there a valid pattern $\mathcal{S} \subset \mathcal{C}$ covering the grid, with $|\mathcal{S}| \leq F$? An instance of the decision rectangular grid cover problem is defined by the system $(G, \mathcal{C}, F)$ and can be encoded by a NMD boolean array using the matrix formulation (2.2).

Below is described a procedure for reducing a vertex cover problem instance $(\mathcal{V}, \mathcal{E}, K)$ into a decision rectangular grid cover problem instance $(G, \mathcal{C}, F)$.

Let the vertices and edges of the graph be ordered as $\mathcal{V} = \{v_0, \ldots, v_{|\mathcal{V}|-1}\}$ and $\mathcal{E} = \{e_0, \ldots, e_{|\mathcal{E}|-1}\}$. Each edge is described by a pair of distinct vertices $e_m = (v_i, v_j)$ with $i < j$.

Let G be a $|\mathcal{E}|$-by-$3|\mathcal{V}|$ rectangular grid. Each row of the grid can be viewed as representing an edge, and each block of three columns represent a vertex, see Figure 2.18. A practical example is shown in Figure 2.24. Three types of rectangular covers are defined on the grid:

- **Column covers**: for each vertex v_i, the column cover representing said vertex is the central column of the block column

$$V_i = \{g_{m,3i+1} : 0 \leq m < M\}$$

see Figure 2.19. Formally, the set of column covers is

$$\mathcal{C}_{\mathcal{V}} = \{V_i : v_i \in \mathcal{V}\}$$

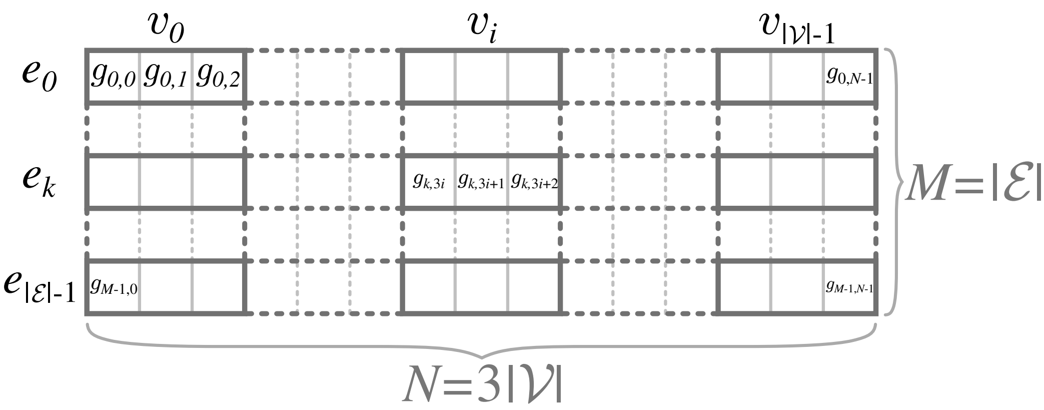

Figure 2.18: Reduction grid of vertex covering into rectangular grid covering

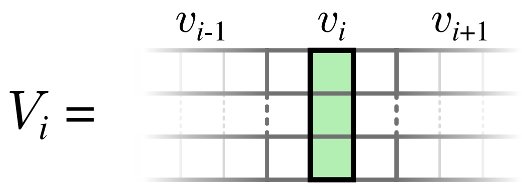

Figure 2.19: Column cover

- **Side-row covers**: for each edge $e_k = (v_i, v_j)$, the left side-row cover includes the beginning of the edge row, up to the left column of the v_i block-column

$$L_k = \{g_{k,n} : 0 \leq n \leq 3i\}$$

and similarly the right side-row cover includes the end of the edge row, starting from the right column of the v_j block-column

$$R_k = \{g_{k,n} : 3j + 2 \leq n < N\}$$

see Figure 2.20. Formally, the set of side-row covers is

$$\mathcal{C}_S = \{L_k : e_k = (v_i, v_j) \in \mathcal{E}\} \cup \{R_k : e_k = (v_i, v_j) \in \mathcal{E}\}$$

Side-row covers are all required to be in a valid pattern, since each is the only cover for either the first or last cell of the row. Thus only the "central part", i.e. $H_k = \{g_{k,n} : 3i + 1 \leq n \leq 3j + 1\}$, of each row still has to be covered.

- **Center-row covers**: for each edge $e_k = (v_i, v_j)$, the "central part" H_k of the edge row is partially covered if the pattern contains a column

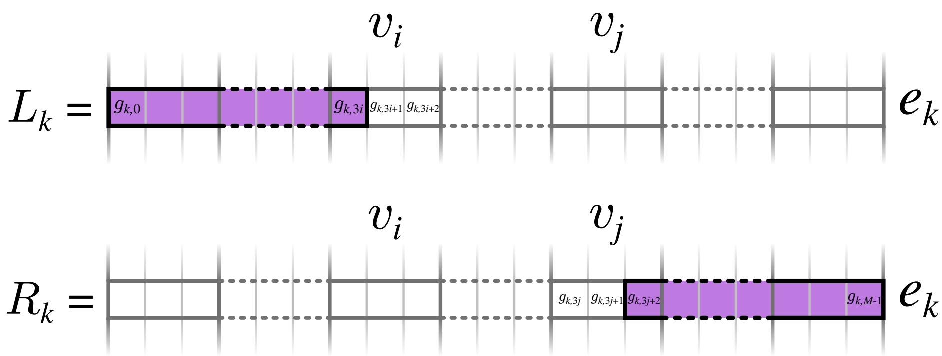

Figure 2.20: Side-row covers

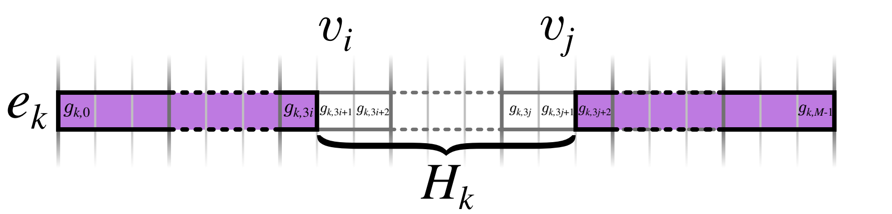

Figure 2.21: The "central part" H_k

cover. The remaining uncovered cells can be covered by the left center-row cover

$$A_k = \{g_{k,n} : 3i + 1 \leq n \leq 3j\}$$

to complement the column cover V_j, or by the right center-row cover

$$B_k = \{g_{k,n} : 3i + 2 \leq n \leq 3j + 1\}$$

to complement the column cover V_i. Thus either combination can cover the row "central part" since $H_k \subset A_k \cup V_j$ and $H_k \subset V_i \cup B_k$. Note that H_k can also be covered by $A_k \cup B_k$, but covering H_k requires two covers in any case. The three possibles configurations are shown in Figure 2.23. The set of all center-row covers is

$$\mathcal{C}_H = \{A_k : e_k =\in \mathcal{E}\} \cup \{B_k : e_k =\in \mathcal{E}\}$$

While center-row covers are not all compulsory, for each row one of the two center-row covers must be in the pattern, as they are the only rectangular covers for cells $\{g_{k,n} : 3i + 2 \leq n \leq 3j\}$ which are between the two columns V_i and V_j.

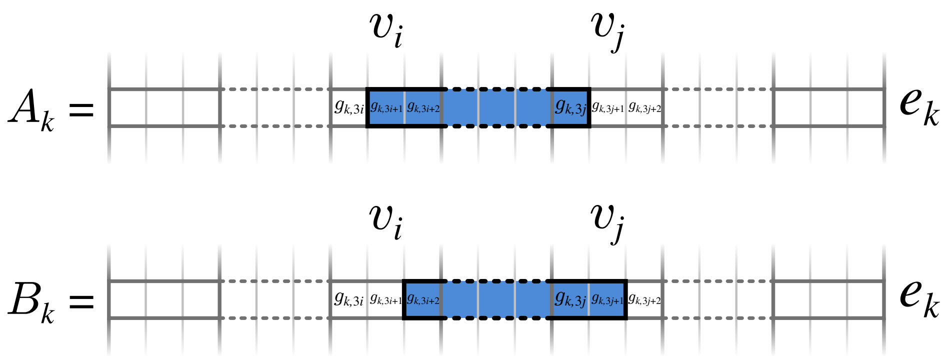

Figure 2.22: Center-row covers

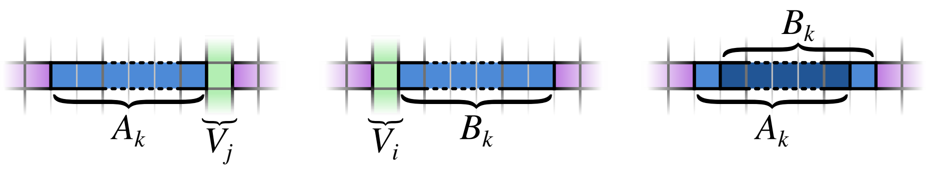

Figure 2.23: The three configurations for covering H_k

So for each row on the grid, the two side-row covers are required. And at least one of the two center-row covers is also required. Thus a valid pattern contains at least $3|\mathcal{E}|$ covers.

Let the set of all rectangular covers be

$$\mathcal{C} = \mathcal{C}_{\mathcal{V}} \cup \mathcal{C}_S \cup \mathcal{C}_H$$

The grid cover instance $(G, \mathcal{C}, 3|\mathcal{E}| + K)$ has a solution if and only if the vertex cover problem instance $(\mathcal{V}, \mathcal{E}, K)$ has a solution. Figure 2.25 present a pair of solutions for the reduction example in Figure 2.24.

Suppose there is a valid solution $\mathcal{U}$ with $|\mathcal{U}| \leq K$ for problem $(\mathcal{V}, \mathcal{E}, K)$. Let $\mathcal{S}_{\mathcal{U}} = \{V_i : v_i \in \mathcal{U}\} \subset \mathcal{C}_{\mathcal{V}}$, thus being the set of column covers corresponding to the vertices in vertex cover $\mathcal{U}$.

For each edge $e_k = (v_i, v_j)$, either $V_i \in \mathcal{S}_{\mathcal{U}}$ or $V_j \in \mathcal{S}_{\mathcal{U}}$. Let E_k be the center-row cover complementing the "central part":

$$E_k = \begin{cases} A_k \text{ if } V_j \in \mathcal{U} \\ B_k \text{ otherwise} \end{cases}$$

and the corresponding row is covered by $L_k \cup E_k \cup V_i \cup R_k$ for some $V_i \in \mathcal{S}_{\mathcal{U}}$. So $\mathcal{S} = \mathcal{S}_{\mathcal{U}} \cup \{E_k : e_k \in \mathcal{E}\} \cup \mathcal{C}_S$ is a valid pattern containing $K + |\mathcal{E}| + 2|\mathcal{E}|$ elements and thus a solution for the grid cover problem instance $(G, \mathcal{C}, 3|\mathcal{E}| + K)$.

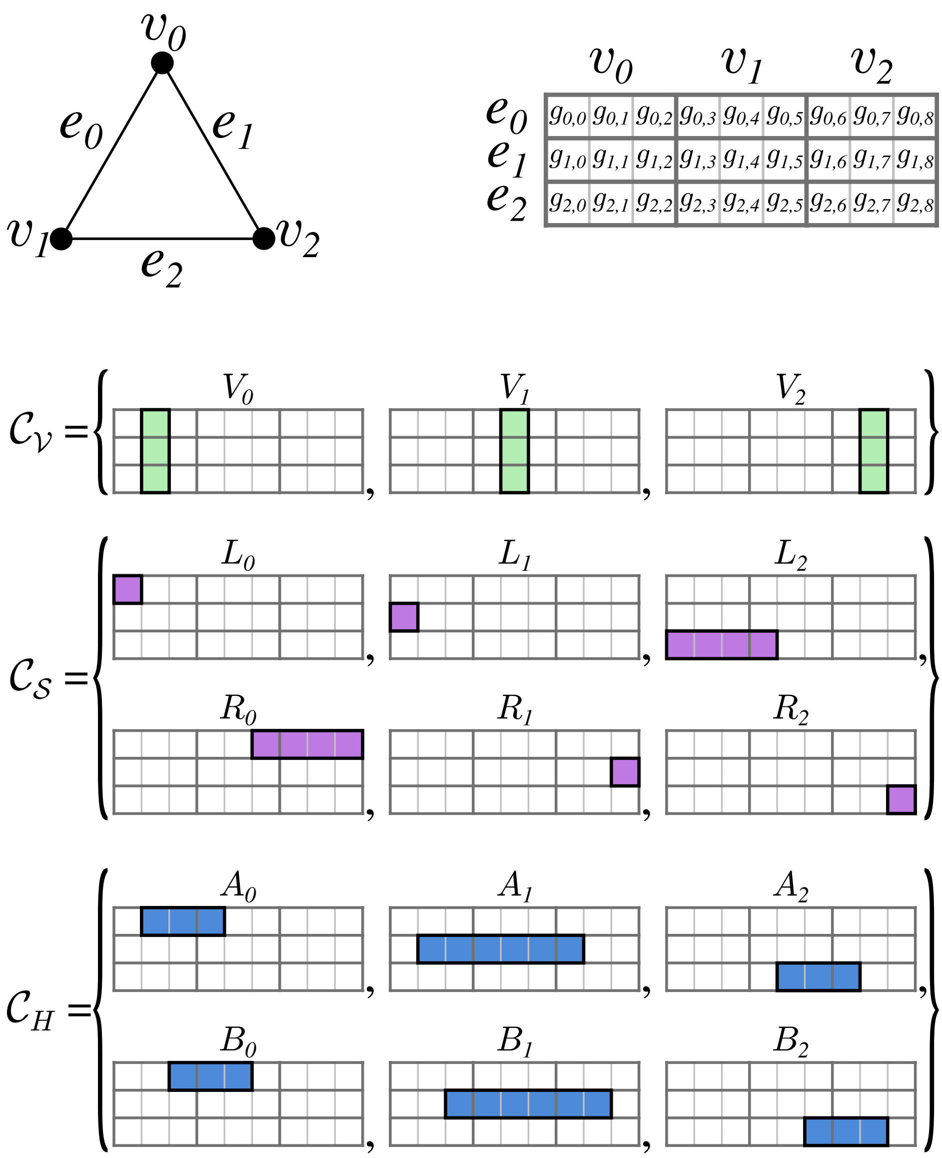

Figure 2.24: Example of a vertex cover reduction to rectangular grid cover, with the input graph (top-left), reduction grid (top-right) and rectangular covers (bottom)

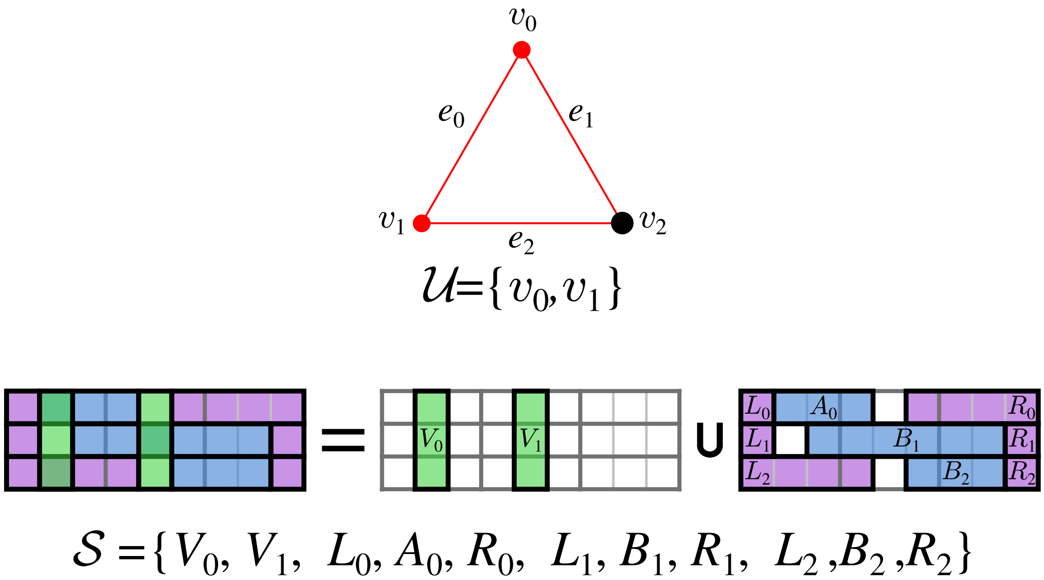

Figure 2.25: Solution for vertex cover problem (left) and corresponding solution for grid cover reduction (right)

Conversely, suppose there is a valid solution $\mathcal{S}$ with $|\mathcal{S}| \leq 3|\mathcal{E}| + K$ for the grid cover problem instance $(G, \mathcal{C}, 3|\mathcal{E}| + K)$.

For each row, there is at least one center-row cover. Suppose there is an edge $e_k = (v_i, v_j)$ whose row is covered by the two center-row covers. Then one of the two covers can be replaced by a column cover: $(\mathcal{S} \leftarrow \mathcal{S} \cup \{V_i\}) \setminus \{B_k\}$ without changing the cardinality of the solution: $|\mathcal{S}| \leq |\mathcal{E}| + 3K$. Iterating this process produces a pattern for which there is exactly one center-row cover per row.

Thus the "central part" H_k of each row is covered by a combination of a center-row cover and a column cover: either $A_k \cup V_j$ or $B_m \cup V_i$. So for each edge $e_k = (v_i, v_j)$, the pattern $\mathcal{S}$ contains a column cover of one its vertex: V_i or V_j, and $\mathcal{U} = \{v_i : V_i \in \mathcal{S} \cap \mathcal{C}_\mathcal{V}\}$ is a valid vertex cover. Furthermore

$$|\mathcal{U}| = |\mathcal{S} \cap \mathcal{C}_\mathcal{V}| = |(\mathcal{S} \setminus \mathcal{C}_H) \setminus \mathcal{C}_S| = |\mathcal{S}| - |\mathcal{E}| - 2|\mathcal{E}| = |\mathcal{S}| - 3|\mathcal{E}| \leq K$$

since $\mathcal{S}$ contains one cover from $\mathcal{C}_H$ per row and all covers in $\mathcal{C}_S$. $\mathcal{U}$ is thus a valid solution for vertex cover problem $(\mathcal{V}, \mathcal{E}, K)$.

So there is a solution for the graph problem $(\mathcal{V}, \mathcal{E}, K)$ if and only if there is a solution for grid problem $(G, \mathcal{C}, 3|\mathcal{E}| + K)$. All that is left is to check that the reduction is polynomial. Suppose the reduced grid problem is encoded using binary matrices. Each rectangular cover has $3|\mathcal{V}||\mathcal{E}|$ boolean elements, and $\mathcal{C}$ contains $|\mathcal{V}| + 4|\mathcal{E}|$ rectangular covers. Generating all covers takes at $O\left(3|\mathcal{V}|^2|\mathcal{E}| + 12|\mathcal{V}||\mathcal{E}|^2\right)$ operations and the problem instance is encoded using $3|\mathcal{V}|^2|\mathcal{E}|+12|\mathcal{V}||\mathcal{E}|^2$ bits. So the reduction is polynomial and rectangular grid covering is NP-hard.

Note that it is possible to compress the problem encoding, similarly to the interval cover problem, since rectangles on a M-by-N grid can be described by four integer values defining the top-left and bottom-right corner. So the problem can be encoded in $2(\log_2(3|\mathcal{V}|) + \log_2|\mathcal{E}|)(|\mathcal{V}| + 4|\mathcal{E}|)$ bits. Since the "uncompressed" problem is already NP-hard, the "compressed" problem is said to be *strongly NP-hard*, i.e. it has a pseudo-polynomial algorithm only if P=NP.

Approximability

There is currently no known methods for solving efficiently NP-hard problems, and there might never be. Thus an important field in optimization is the research of polynomial approximation algorithms, which returns in polynomial time a valid non-optimal solution, however guaranteed to be within a certain ratio of the optimal cost for the problem. For a minimization problem, an algorithm is said to be an α-approximation algorithm if it returns a solution with cost F_{apx} such that $F_{apx} \leq \alpha F_{opt}$ with F_{opt} the cost of an optimal solution.

Unfortunately, covering problems are not easy to approximate in general. The general set cover problem is known to be log-approximable [13], using the greedy method which return a solution whose value is at most $\log(|G|)$ times the optimal cost. But it is also log-APX-complete [16], meaning set cover problem is at least as hard as all other problems which are log-approximable.

Specific cases of the set cover problem can achieved better approximations however. The minimum vertex cover problem has a 2-approximation algorithm [26]. In fact, all covers problem with a constant *frequency parameter* f can be f-approximated using a primal-dual algorithm [31]. The frequency factor is defined as

$$f = \max_i |\{j : g_i \in C_j \wedge C_j \in \mathcal{C}\}|$$

and represents the maximum number of times an element appears in the collection of available sets, or using radar terminology the maximum number of overlaps of dwells discrete cover. This value is however not bound in general, and thus the primal-dual approach does not guarantee constant approximation ratio in all cases.

The minimum vertex cover problem is also known to be APX-complete, meaning at least as hard as all problems approximable in constant ratio.

While the previous reduction of vertex covering to rectangular grid covering is polynomial, it is not an approximation-preserving reduction:

Consider a graph $(\mathcal{V}, \mathcal{E})$, for which a minimum vertex cover has optimal cardinal K_{opt}. The decision vertex cover instance $(\mathcal{V}, \mathcal{E}, K_{opt})$ is true and the decision instance $(\mathcal{V}, \mathcal{E}, K_{opt} - 1)$ is false. The grid cover problem $(G, \mathcal{C})$ obtained via the reduction presented previously has thus an optimal solution with cost $F_{opt} = 3|\mathcal{E}| + K_{opt}$.

Suppose there is an α-approximation algorithm for the grid cover problem, which returns an approximated solution with cost $F_{apx} \leq \alpha F_{opt} = \alpha(3|\mathcal{E}| + K_{opt})$. From this solution, an approximation vertex cover for the original problem can be computed by replacing and removing center-row and side-row covers, as has been done at the end of the NP-hardness proof. The approximation vertex cover has a cost

$$K_{apx} = F_{apx} - 3|\mathcal{E}| \leq \alpha(3|\mathcal{E}| + K_{opt}) - 3|\mathcal{E}| = \alpha K_{opt} + (\alpha - 1)3|\mathcal{E}|$$

But this cost can be arbitrarily high since a graph with a size-bounded optimal vertex cover can have an arbitrarily high number of edges, for example a star graph S_k. So there is no ratio β such that $(\alpha - 1)3|\mathcal{E}| \leq \beta K_{opt}$, and the reduction is not approximation-preserving. The exact approximability of the rectangular grid covering remains an open question, though the problem is at worst log-approximable.

2.5.2 Connected grid cover problem

The radar model with connected dwell discrete covers is a more general case of the previous problem where the set of available discrete covers can contains any kind of connected shape, according to the definition given in 2.2.3. Since rectangular shapes are valid connected shape, the problem is immediately NP-hard, as any problem instance of rectangular grid covering is a valid instance of connected grid covering. However, an alternate reduction from the general set cover problem can be made.

Let $(G, \mathcal{C}, K)$ be an instance for the general set cover problem. Let G' be a 2-by-$|G|$. For each cover $C_j \in \mathcal{C}$, let

$$C'_j = \{g'_{0,i} \in G' : g_i \in C_j\} \cup \{g_{1,0}, \ldots, g_{1,|G|}\}$$

such that the first line of cover C'_j replicates C_j, while the second line of C'_j contains all elements on the second line of G', see Figure 2.26, ensuring that C'_j is a connected set. Let $\mathcal{C}' = \{C'_j : C_j \in \mathcal{C}\}$.

Suppose $\mathcal{S} \in \mathcal{C}$ is a solution for set cover problem instance $(G, \mathcal{C}, K)$ and let $\mathcal{S}' = \{C'_j : C_j \in \mathcal{S}\}$. For any detection cell $g_i \in G$, there is $C_j \in \mathcal{S}$ such that $g_i \in C_j$ and thus $g'_{0,i} \in C_j$, furthermore $g'_{1,i} \in C'_j$.

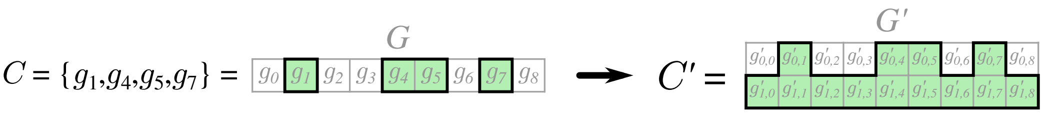

Figure 2.26: Reduction from general set covering to connected grid covering

So $\mathcal{S}'$ is valid solution with $|\mathcal{S}'| = |\mathcal{S}| = K$ for the connected grid cover problem instance (G', C', K). In a similar manner, it is straightforward to show that if $\mathcal{S}' \subset \mathcal{C}'$ is valid solution for a problem instance (G', C', K), then $\mathcal{S} = \{C_j : C'_j \in \mathcal{S}'\}$ is a valid solution for (G, C, K).

Thus $\mathcal{S}$ is a solution for (G, C, K) if and only if $\mathcal{S}'$ is a solution for $(G', \mathcal{C}', K)$ and the two problems are computationally equivalent. This reduction keeps the same cost function for both problems, and is stronger than for the previous reduction of vertex covering to rectangular grid covering, as it preserves approximation properties. Thus connected grid covering is NP-hard, and also log-APX-complete, like general set covering [16].

2.6 Branch-and-bound

In practice, NP-hard optimization problems such as general set covering encountered in industrial settings are often solved by *branch-and-bound*, a combinatorial optimization paradigm whose principle is to explore the decision space searching for a good solution. Its key feature is to avoid exhaustive enumeration of entire branches of the space by bound estimation, hence its name [32]. Despite lacking provably good worst-case theoretical complexity, branch-and-bound generally performs efficiently on practical cases.

2.6.1 Description

The decision space of all possible solutions can be represented as a finite binary tree with depth p, each node representing the value choice of an integer variable, see Figure 2.27. Each end leaf represents a solution for the integer program. The number of possible solutions is finite, but grows exponentially and is usually huge: in the case of a cover problem with D candidate covers, there is 2^D possible pattern solutions.

Exploring the entire tree is computationally infeasible in reasonable time. However it is possible at each node to estimate a lower bound of the node sub-tree best solution, by solving its linear relaxation with methods previously described in 2.3.3. Knowing their lower bound, it is possible to avoid exploring certain subsets:

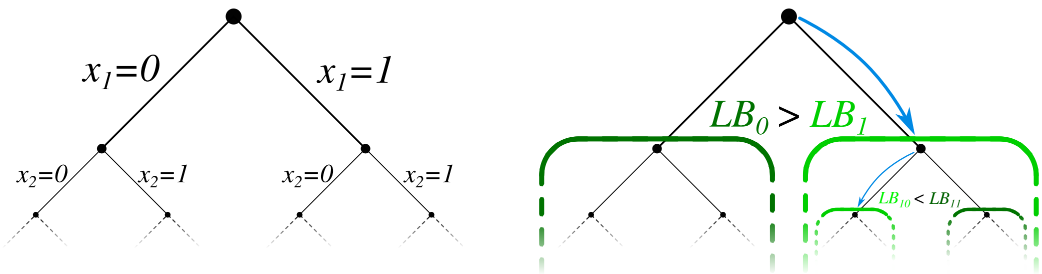

Figure 2.27: Finite tree of solutions (left) and branch-and-bound method (right)

- Branching: Each branch at the current node (with depth $i-1$) correspond to a chosen value, 0 or 1, for the next variable x_i. In each branch, x_i is no longer a variable but a parameter. The current problem is thus divided into 2 smaller sub-problems, each considering a different value for x_i and each having one less variable.
- Bounding: The current problem is relaxed into a linear program, whose solution is a lower bound of the current problem best solution. Depending on the lower bound value, the node sub-tree will be explored next (if it is the most promising branch), later (if there is a more promising branch), or never (if a better solution has already be found in another branch).

Defining what a promising branch is a difficult question, a lower bound is not necessarily better since deeper nodes may have higher bounds while being closer to optimal solutions. Integer programming solvers usually rely on various heuristics to define the exploration strategy and improve bound estimations.

2.6.2 Algorithm

A description of the branch-and-bound method is given below. Algorithm 5 details the corresponding pseudo-code. Each node in the tree can be described by the sequence of choices leading to this node from the root node

$$N = (x_1, x_2, \ldots, x_d)$$

and each node has two children $N_0 = (x_1, \ldots, x_d, 0)$ and $N_1 = (x_1, \ldots, x_d, 1)$. At each node N explored, the first d variables $(x_1, x_2, \ldots, x_d)$ are set, and a linear relaxation of the problem is solved with respect to the remaining free variables $(x_{d+1}, \ldots, x_D)$, then add N to the list of nodes to explore.

The algorithm can be summarized by the following steps:

0. Initialization:
Initialize the list of node to explore with the root node.

1. Exploration:
Pop next node to explore from the list of nodes and solve its linear relaxation.

2. Bounding:
If the current node relaxation value is less than the current best solution found, proceed to Step 3, otherwise, drop current node and go back to Step 1.

3. Update:
If the current node relaxation is an integral solution, then its an improving solution (note that an end leaf always yield an integral solution). Update best current solution and proceed to Step 1.
Otherwise:

4. Branching:
Compute the current node children. For each child, check if the descendants contains a valid solution (this can be done by summing covers already used by the parent, the cover of the child node if used, and covers available to the descendants). If the child node is valid, add it to the list of node to explore. Proceed to Step 1.

This very generic description is just a presentation of the general idea of the method. Efficient implementations of the branch-and-method usually combined several techniques such as cutting planes, diving heuristics and local branching to improve bounds estimation and speed.

2.6.3 Example

The branch-and-bound method is applied on the example from Figure 2.2, described by the integer program (2.3), see Figure 2.28:

- $\mathcal{N} = \{\}$, $\mathbf{x}_{best} = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$, $f_{best} = \mathbf{T}^T \cdot \mathbf{x}_{best} = 13$:
Solving the root relaxation yields the linear solution $(0\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2})$ with cost $\frac{11}{2} \leq 13$. Root node children (0) and (1) are feasible, and thus added to the exploration list $\mathcal{N} := \{(0), (1)\}$

Algorithm 5 Branch-and-bound

```
% LP_SOLVE is the relaxation subroutine called during branching
function LP_SOLVE(N)
    (x_1, ..., x_{d-1}) := N                                   ▷ node N sets first d − 1 variables
    (x_d, ..., x_D) := argmin{∑_{j=d}^{D} T_j x_j : A · x ≥ 1}   ▷ optimize free variables
    return x_L := (x_1, ..., x_d, x_{d+1}, ..., x_D)
end function

% Initialization
N_root = ()
𝒩 := {N_root}                                                ▷ start with root node
x_best := x_F = (1 ··· 1)                                    ▷ best current solution (default is x_F)

% Exploration
while 𝒩 is not empty do
    N := pop(𝒩)                                              ▷ take next node in 𝒩
    x_L := LP_SOLVE(N)                                       ▷ solve node relaxation

    % Bounding
    if T^T · x_L < T^T · x_best then                         ▷ explore node N if improvement is possible

        % Update
        if x_L ∈ {0, 1}^D then                               ▷ check if x_L is an integral solution
            x_best := x_L
        else
            (x_1, ..., x_d) := N

            % Branching
            for x ∈ {0, 1} do                                ▷ compute children of node N
                N_c := (x_1, ..., x_d, x)
                if A · (x_1 ··· x_d x 1 ··· 1)^T ≥ 1 then    ▷ check child feasibility
                    𝒩 := 𝒩 ∪ {N_c}                           ▷ add child to candidate list
                end if
            end for
        end if
    end if
end while
return x_best
```

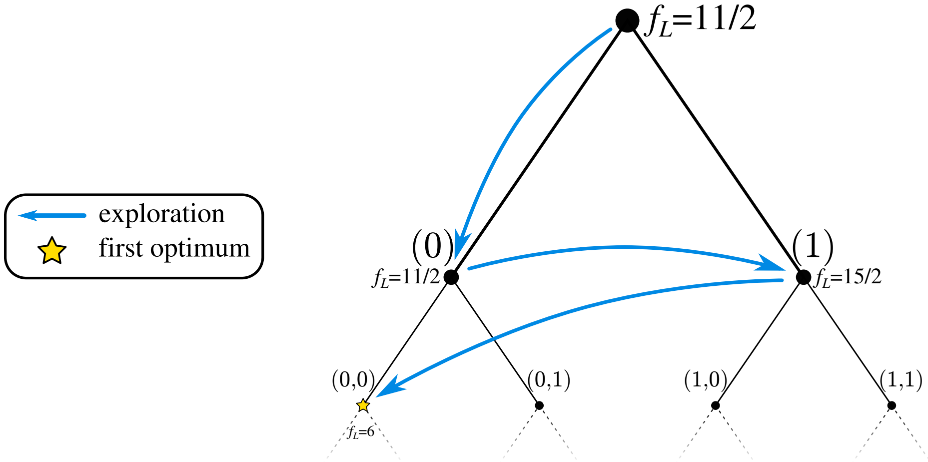

Figure 2.28: Branch-and-bound application example

- $\mathcal{N} = \{(0),(1)\}$, $\mathbf{x}_{best} = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$, $f_{best} = \mathbf{T}^T \cdot \mathbf{x}_{best} = 13$:
  Relaxation of (0) yields the same linear solution $(0\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2})$ with cost $\frac{11}{2}$. We add the children $(0,0)$ and $(0,1)$ to the exploration list $\mathcal{N} := \{(1),(0,0),(0,1)\}$

- $\mathcal{N} = \{(1),(0,0),(0,1)\}$, $\mathbf{x}_{best} = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$, $f_{best} = \mathbf{T}^T \cdot \mathbf{x}_{best} = 13$:
  Relaxation of (1) yields the linear optimal solution $\mathbf{x}_L = (1\ 0\ 1\ 1\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2})$ with cost $\frac{15}{2} < 13$. We add the children $(0,0)$ and $(0,1)$ to the exploration list $\mathcal{N} := \{(1,0),(1,1)\}$

- $\mathcal{N} = \{(0,0),(0,1),(1,0),(1,1)\}$, $\mathbf{x}_{best} = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$, $f_{best} = \mathbf{T}^T \cdot \mathbf{x}_{best} = 13$:
  Relaxation of $(0,0)$ yields the linear optimal solution $\mathbf{x}_L = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)$ with cost $6 < 13$. $\mathbf{x}_L$ is an integral solution, thus we update the best current solution $\mathbf{x}_{best} := \mathbf{x}_L$; $f_{best} := 6$.

At this point, it can be deduced that $\mathbf{x}_{best}$ is an integer optimal solution. The root relaxation has linear optimal cost $\frac{11}{2}$. By bounding, any integer solution has an integer cost greater than the linear optimal cost $\frac{11}{2}$, so greater than $6 = \lceil \frac{11}{2} \rceil$. This suffices to prove the optimality of $\mathbf{x}_{best} = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)$ for the integer program (2.3).

2.6.4 Just-in-time criteria

One of the most interesting features of the branch-and-bound method from an operational point of view is the possibility to use a “just-in-time” criteria.

For example, a radar system with an embedded computer must optimize its cover just before a mission start. However, it only has five minutes to perform the optimization. A "just-in-time" criteria impose a time limit ensuring that even if the optimum has not been reached, the algorithm will return the best solution it found in the available lapse of time. Another advantage is the lower bound of the optimal cost provided by linear relaxation:

$$B_{\mathcal{N}} = \min\{\mathbf{T}^T \cdot \mathbf{x}_L \ : \ \mathbf{x}_L = \text{LP\_SOLVE}(N), N \in \mathcal{N}\}$$

thus during the computation, the method always has an confidence interval for the optimal solution value, above the lower bound but below the current best value:

$$B_{\mathcal{N}} \leq \mathbf{T} \cdot \mathbf{x}_{opt} \leq \mathbf{T} \cdot \mathbf{x}_{best}$$

Knowing the lower bound, the *(worst-case) relative optimality gap* is:

$$\Delta_{opt} = \frac{\mathbf{T} \cdot \mathbf{x}_{best} - B_{\mathcal{N}}}{B_{\mathcal{N}}}$$

which give as a percentage the best possible gain for an optimal solution relatively to the current best solution. The pseudo-code modifications required to account for a time limit and provide the current lower bound are described in Algorithm 6.

Algorithm 6 Just-in-time branch-and-bound

```
% Exploration
current_time := time()                                        ▷ Get current time
while 𝒩 is not empty AND current_time ≤ time_limit do
    ...
end while
return 𝒳_best, B_𝒩
```

In practice, if the algorithm has a broad choice of available covers, it will find very quickly a good quality solution. Typically within $\leq 10\%$ of relative optimality gap. However closing those last percents to reach the optimal solution can be difficult. Because the decision space is often huge, the algorithm spends a long time crossing out possibilities. In some case even, the algorithm finds quickly the optimal solution, and spends a long time proving its optimality.

Chapter 3

Radar search pattern optimization

Multi-function radars usually perform multiple tasks simultaneously, such as scanning, target tracking and identification, clutter mapping, etc. [33, 34, 35, 36]. Electronic scanning and numerical processing allow dynamical use of beam-steering, beam-forming, dwell scheduling and waveform processing to adapt to operational requirements. As complex situations can result in system overload, multi-function radars must optimize resources allocation to ensure robust detection. Optimization of the radar search pattern minimizes the required time-budget for radar scanning, thus freeing resources for other tasks.

In the past several works have explored various approaches for optimization of the radar search pattern: [37, 38] optimized scanning by tiling identical pencil beams over the surveillance space, [39] developed adaptive activation strategies on a pre-designed radar search pattern. Those approaches however do not fully use active radars capabilities to dynamically perform beam-forming. A similar problem is wireless network covering: for a given base station and given clients, ensure connection for all clients using a minimal numbers of directive antenna [40, 41]. Radar search covering and wireless network covering have similar underlying mathematical structures with both being covering problems.

3.1 General optimization problem

A *radar search pattern* is a collection of dwells ensuring detection over the surveillance space. An optimal radar search pattern achieves detection using a minimum time-budget. The surveillance space $\mathcal{A}_S$ defines the azimuth-

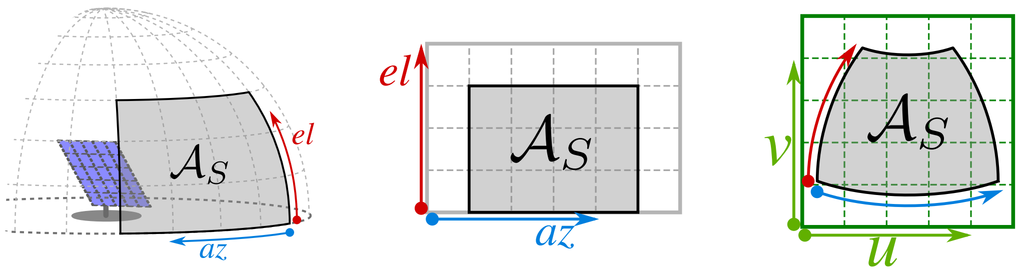

Figure 3.1: Surveillance space in 3D (left), in azimuth/elevation (center), in direction cosines (right)

elevation scanning range, see Figure 3.1:

$$\mathcal{A}_S = [az_{min}, az_{max}] \times [el_{min}, el_{max}] \in\subset [-\frac{\pi}{2}, \frac{\pi}{2}] \times [0, \frac{\pi}{2}]$$

where az and el are respectively the azimuth and elevation angles in radians.

3.1.1 Detection constraint

The radar search pattern must ensure detection for a given mission with requirements defined by several parameters:

- σ is the radar cross-section of the target type.
- $R_c : \mathcal{A}_S \to \mathbb{R}^+$ is the desired detection range. In general, the desired detection range is defined by height $H_{\min}$ and distance $D_{\min}$, see Figure 3.2:
$$R_c(az, el) = \begin{cases} D_{\min} & \text{if } el \leq \operatorname{asin}\left(\frac{H_{\min}}{D_{\min}}\right) \\ \frac{H_{\min}}{\sin(el)} & \text{otherwise} \end{cases}$$
- $i \in \{0, \cdots, 4\}$ is the Swerling model [8].
- $P_d \in]0, 1[$ is the desired detection probability and $P_{fa} \in]0, 1[$ is the desired false alarm probability.

The radar search pattern ensures detection if for each direction $(az, el) \in \mathcal{A}_S$, the radar search pattern contains at least one dwell capable of detecting a target with radar cross-section σ at range $R_c(az, el)$ with at least detection probability P_d and at most false alarm probability P_{fa}.

Each dwell has a processing time, the time duration of its associated waveform, during which the radar cannot perform other action, whether emitting another dwell or accomplishing tracking tasks. The radar search pattern

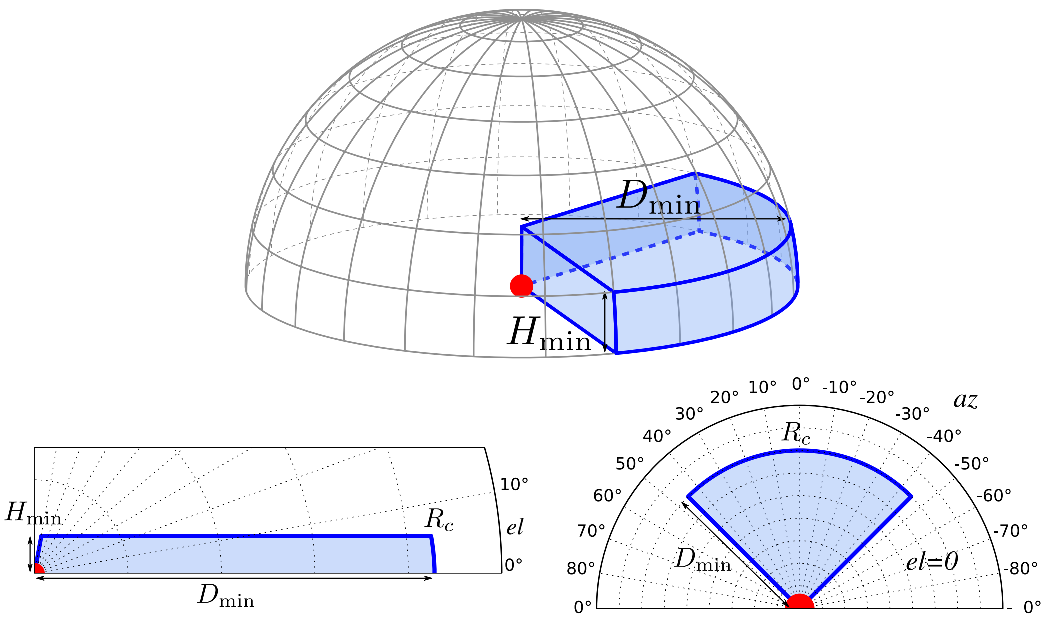

Figure 3.2: Desired detection range (top), azimuth cut (bottom-left), elevation cut (bottom-right)

time-budget is the sum of all its dwells associated waveform duration, thus the time taken to perform the entire radar search pattern. The optimization problem is to find a radar search pattern ensuring the detection constraint for a minimal time-budget.

3.1.2 Radar system parameters

To achieve the detection requirements described previously, an available radar system is described by the following parameters:

- Radar peak power : P_p
- Phased array dimensions parameters : $M,\ N,\ d_x,\ d_z$

and this system has access to database of waveforms

$$\mathcal{W} = \{w_1, \ldots, w_P\}$$

each waveform $w \in \mathcal{W}$ being described by:

- Its duration T_w
- Its average dutycycle f_w

- Its carrier wavelength λ_w
- Its required signal-to-noise ratio s_w for desired detection and false alarm probabilities, which can either come from measurements or simulations, or either be computed using the waveform model described in 1.5, knowing the number of bursts N_b and the detection threshold K_b in the waveform.

3.1.3 Digital beamforming processing limit

A dwell d ensures detection over the surveillance space subset

$$\mathcal{A}_d = \{(az, el) \in \mathcal{A}_S : R_d(az, el) \geq R_c(az, el)\} \tag{3.1}$$

whose area is limited by the radar maximum digital beamforming scanning area $A_{\max}$

$$A_d = \iint_{\mathcal{A}_d} dudv \leq A_{\max}$$

3.1.4 Problem statement

Finding a radar search pattern $\mathcal{S}_{opt}$ ensuring the detection constraint over the surveillance space with minimal time-budget is a minimization problem under constraints:

$$\min \sum_{0 \leq j \leq J} T_{w_j} \tag{3.2a}$$

$$s.t.\ \mathcal{S} = \{d_j, 0 \leq j \leq J\},\ \ J \in \mathbb{N} \tag{3.2b}$$

$$\mathcal{A}_S \subset \bigcup_{d \in \mathcal{S}} \mathcal{A}_d \tag{3.2c}$$

$$\forall d \in \mathcal{S},\ \ A_d = \iint_{\mathcal{A}_d} dudv \leq A_{\max} \tag{3.2d}$$

The problem amounts to finding a radar search pattern $\mathcal{S}$ containing a finite number of dwells (3.2b), validating detection constraint over the entire surveillance space for the given mission (3.2c), with each dwell processable at reception (3.2d), and using minimal radar time-budget (3.2a).

3.2 Problem discrete approximation

The general optimization problem is difficult to solve for several reasons:

- continuous variables in the phase-amplitude illumination law of each dwell radiation pattern mixed with discrete variables for each dwell waveform choice.
- the number of variables is not set, as it depends on the number of dwells, introducing a “meta-variable”.
- the desired detection range R_c is not generally a convex function.

It is thus a non-convex mixed optimization problem, with potentially a large varying number of variables. A more sensible way to tackle this problem is to approximate it as a combinatorial set cover problem, since it intuitively possesses a similar structure as a covering problem.

The approximation turning the general problem into a combinatorial set cover problem relies on two assumptions:

- The use of discrete grid representing the surveillance space. In the original problem, there is no quantification of the surveillance space, which is a continuous set.
- The restriction to rectangular radiation patterns. A phased array antenna can theoretically produce all sorts of beam-shaped radiation patterns, and the set of possible patterns is in fact continuous. This is impractical for a combinatorial formulation, which requires a finite set sampled amongst all possibilities. Choosing this set as the collection of all possible rectangular patterns offers a broad choice of covering while avoiding combinatorial explosion of oversampling.

Under those assumptions, the procedure for approximating a solution to the general problem can be divided into three steps:

- space quantification: the definition of finite bi-dimensional grid covering and representing the surveillance space.
- pattern synthesis: the generation of a collection of rectangular candidate dwells ont the grid.
- combinatorial optimization: the selection of an optimal subset among the rectangular candidate dwells.

3.2.1 Detection grid

The surveillance space in direction cosines coordinates is approximated by a finite bi-dimensional M-by-N regular grid, see Figure 3.3. On this grid,

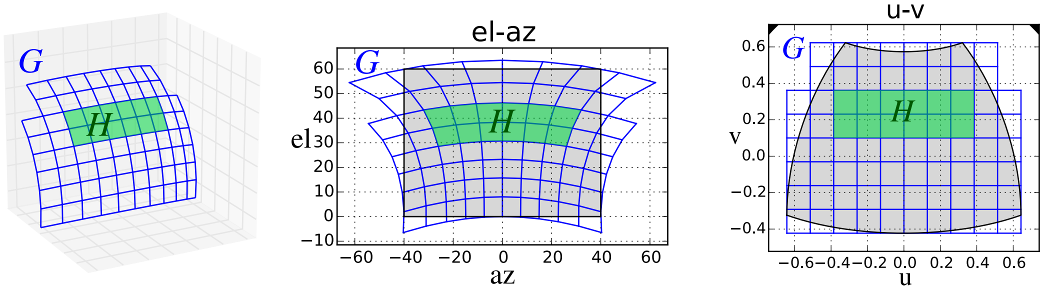

Figure 3.3: Detection grid G and a rectangle H in 3D (left), in azimuth/elevation (middle), in direction cosines (right)

the detection constraint is considered on each cell, with a finite number of cells, instead of working on the continuous set of possible azimuth-elevation directions.

Let $[u_{\min}, u_{\max}] \subset [0, 1]$ and $[v_{\min}, v_{\max}] \subset [0, 1]$ be the radar scanning range in direction cosines coordinates on the surveillance space. Let $M \in \mathbb{N}^*$ and $N \in \mathbb{N}^*$ define the desired grid resolution. Then the grid nodes are computed by :

$$\begin{array}{lll} u_0 = u_{\min}, & u_N = u_{\max} & u_n = u_0 + n\left(\frac{u_N - u_0}{N}\right) \\ v_0 = v_{\min}, & v_M = v_{\max} & v_m = v_0 + m\left(\frac{v_M - v_0}{M}\right) \end{array} \tag{3.3}$$

Any rectangle H on grid G can be characterized by its upper left corner (u_n, v_m) and its lower right corner (u_q, v_r) on the grid, such that $0 \leq n < q \leq N$ and $0 \leq m < r \leq M$, see Figure 3.3. The number of possible rectangles on G is bounded by

$$\frac{MN(M+1)(N+1)}{4}$$

3.2.2 Pattern synthesis

Let H be a rectangle on grid G, characterized by nodes (u_n, v_m) and (u_q, v_r). The ideal radiation pattern covering H is

$$g_H(u, v) \propto \begin{cases} L_s(u, v)^2 \left\{\frac{R_c(u,v)^4 s_w}{\sigma}\right\} & \text{if } u_n \leq u \leq u_q \text{ and } v_m \leq v \leq v_r \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

up to a constant factor, as the array antenna feeds are normalized. This radiation pattern fits the maximum of ideal energetic distributions for all mission detection constraints. This type of ideal pattern is usually infeasible on a real antenna, because it features discontinuities on the rectangle edges,

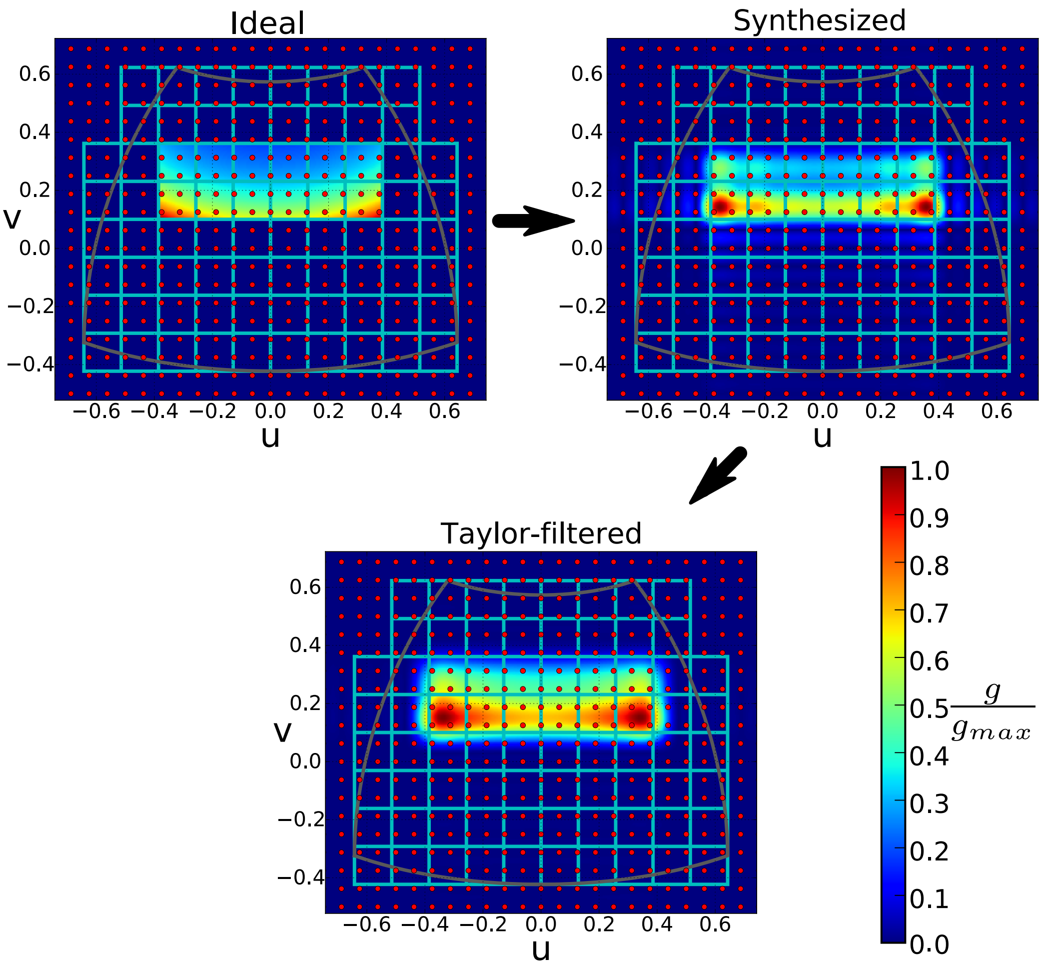

Figure 3.4: Ideal radiation pattern (top-left), synthesized radiation pattern (top-right) and synthesized radiation pattern after Taylor filtering (bottom), with synthesis sampling points in red.

see Figure 3.4. The radiation pattern is the Fourier transform of the antenna illumination law, see (1.3). A discontinuous radiation pattern would require an infinitely large array antenna, for the same mathematical reasons that a discontinuous time signal has an infinite spectrum.

A feasible radiation pattern $\hat{g}_H$ can be synthesized by applying a bi-dimensional Woodward-Lawson sampling method to the ideal pattern g_H, adapted from the one-dimensional method described in [2, 3]. Using sampled values of the desired pattern at evenly-spaced sampling points (in red), the method synthesizes a feasible pattern that is guaranteed to hold the same values at the sampling points, see Figure 3.4. The sampling points form a K'-by-L' grid with nodes (u_l, v_k), $0 \leq l < L'$, $0 \leq k < K'$ (note that this

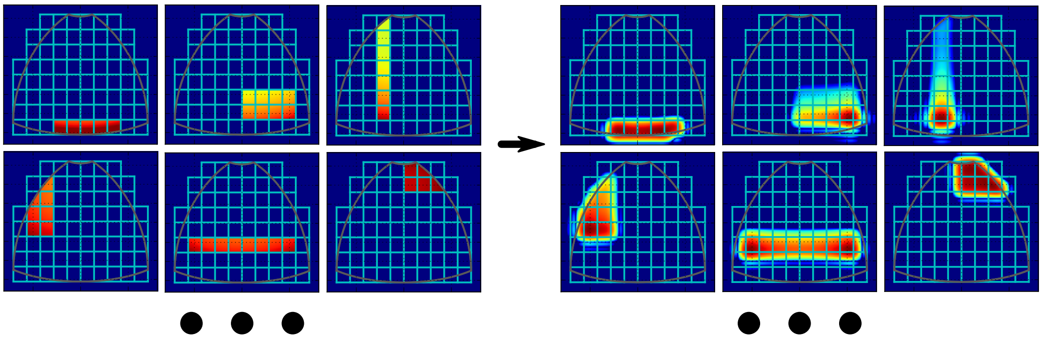

Figure 3.5: Pattern synthesis applied to a database of ideal rectangular radiation patterns

grid has no relation to detection grid G) with:

$$\begin{array}{llll} L' & = 2\lfloor \frac{L}{2} \rfloor + 1, & u_l & = \frac{2l+1-L'}{L} \\ K' & = 2\lfloor \frac{K}{2} \rfloor + 1, & v_k & = \frac{2k+1-K'}{K} \end{array} \tag{3.5}$$

The number of sampling points along one dimension is the closest rounded-up odd number to the number of radiating elements on the same axis. The feeds of the feasible pattern are computed using the ideal pattern values at the sampling points:

$$\hat{a}_{k,l} = \frac{1}{KL} \sum_{k'=0}^{K'} \sum_{l'=0}^{L'} g_H(u_{l'}, v_{k'}) e^{-j\pi(kd_y v_{k'} + ld_x u_{l'})/\lambda}$$

The feeds are normalized: $\hat{a}_{k,l} \leftarrow \hat{a}_{k,l} / \max_{k,l}\{\hat{a}_{k,l}\}$ and Taylor filtering is used for decreasing sidelobes and Gibbs oscillations. From the feeds, the feasible pattern can be computed using (1.3).

Applying this synthesis procedure to all possible rectangles on grid G, with area A_H inferior to the maximum digital beamforming scanning area $A_{\max}$ described in 3.1.3, generates a collection of processable radiation patterns, as shown in Figure 3.5:

$$\mathcal{T} = \{\hat{g}_H : H \subset G \land A_H \leq A_{\max}\}$$

Other synthesis methods based on least square optimization [42], genetic algorithms [43] and alternating projections [44] are also compatible with this approach.

3.3 Set cover problem formulation

The set of candidate dwells $\mathcal{D}$ can be computed as the Cartesian product of $\mathcal{T}$, the set of synthesized radiation patterns, and $\mathcal{W}$, the set of available

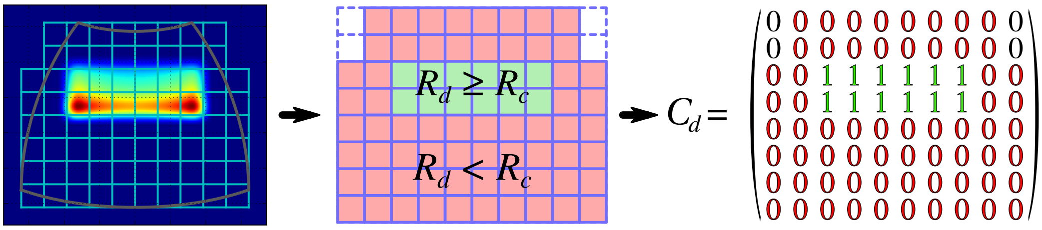

Figure 3.6: Computation of discrete covers for one dwell on two scanning missions

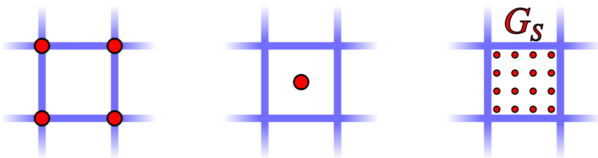

Figure 3.7: Corner sampling (left), center sampling (middle), subgrid oversampling (right), with sampling points in red

waveforms :

$$\mathcal{D} = \mathcal{T} \times \mathcal{W} = \{(g_t, w), g_t \in \mathcal{T}, w \in \mathcal{W}\} = \{d_1, \cdots, d_p\}$$

3.3.1 Discrete cover computation

The discrete cover of each dwell is a boolean representation of the dwell detection on the grid. It indicates the cells on which the dwell validates the detection constraint, see Figure 3.6.

The discrete cover correspond to a “sampling” of the dwell detection on the grid. Various sampling schemes can be used for computing the discrete cover C_j of a dwell $d_j \in \mathcal{D}$, see Figure 3.7:

- sampling of the cell corners (which are the grid nodes):

$$C_j(m, n) = \bigwedge_{(u,v) \in \{u_n, u_{n+1}\} \times \{v_m, v_{m+1}\}} (R_j(u, v) \geq R_c(u, v))$$

  with $\bigwedge$ the boolean AND operator.

- sampling the cell center:

$$C_j(m, n) = \left(R_j\left(\frac{u_n + u_{n+1}}{2}, \frac{v_n + v_{m+1}}{2}\right) \geq \left(\frac{u_n + u_{n+1}}{2}, \frac{v_n + v_{m+1}}{2}\right)\right)$$

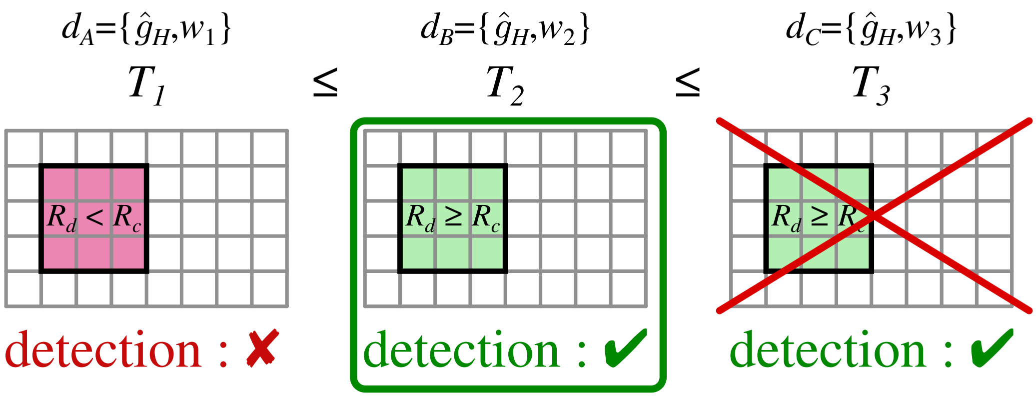

Figure 3.8: Waveform selection, with dwell d_B achieving detection in shortest time with selected waveform w_2

- oversampling a smaller subgrid G_S inside the cell:

$$C_j(m,n) = \bigwedge_{(u,v)\in G_S} (R_j(u,v) \geq R_c(u,v))$$

where R_c is the desired detection range, and R_j is dwell d_j detection range, computed by the radar equation as described in 1.6.

Subgrid oversampling is the most accurate scheme for ensuring that the cell is entirely covered but has a higher computational cost, since each sampling point requires the computation of radar equation with the dwell parameters. In practice, corner sampling usually offers a good compromise between accuracy and computational cost.

3.3.2 Waveform selection

Two dwells using the same pattern but different waveforms may cover the same area, and thus result in the same discrete cover, but with different costs, see Figure 3.8. Quantitatively, one the two dwells dominates the other on the mission, as it validates the same constraint in shorter time. In such case, the costlier dwell can be removed from the set of candidate dwells, because any solution using that dwell could be improved by replacing by the less expensive dwell. This is a form of variable elimination, also called *column reduction*. A more general column reduction method is presented in 4.2.1.

3.3.3 Combinatorial cover problem

Finding a radar search pattern validating the detection constraint over the surveillance space amounts to finding a subset among candidate dwells whose

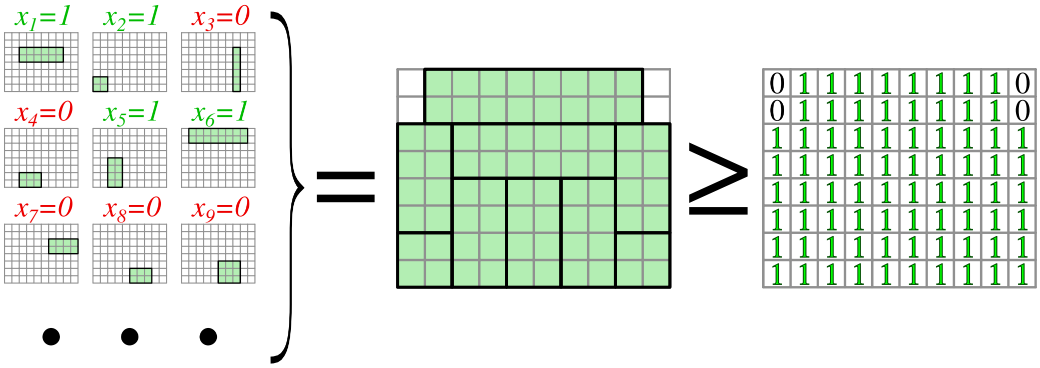

Figure 3.9: The set of available discrete covers with the chosen number of scan for each cover (left), the sum of the chosen discrete covers (middle) and the desired scan update rate for each cell (right)

sum of discrete covers cover the entire grid G, with each cell $G(m, n)$ being covered by at least one dwells, see Figure 3.9. And each discrete cover has an associated cost T_w, also noted T_j in the following, which is its dwell waveform duration.

This covering problem corresponding exactly to the rectangular grid cover problem from 2.5.1, and can be solved by the branch-and-bound method described in 2.6.

3.4 Simulation example

The approximation procedure described previously was applied to a study case. The radar array antenna has 20×20 half-spaced radiating elements. The grid G is laid on a 20×20 lattice. The radar has two available waveforms $\mathcal{W} = \{w_1, w_2\}$, with a long waveform w_1 and a short waveform w_2. The approximation procedure produced 32810 feasible dwells. The detection grid contains 326 detection cells. The corresponding integer program has 32810 variables and 326 detection constraints.

The integer program is computed using Python, and optimization is done with CPLEX [45]. Total computation time for finding one optimal solution is 24 seconds on an i7-3770@3.4GHz processor with a random-access memory (RAM) usage of 450 megabytes.

The obtained solution uses 16 dwells to cover the surveillance area, as shown in Figure 3.10. Dwells covering low elevations have long waveforms (in red), as they must achieve a higher detection range, and thus require more energy, while dwells at high elevations use the short waveform (in blue). The

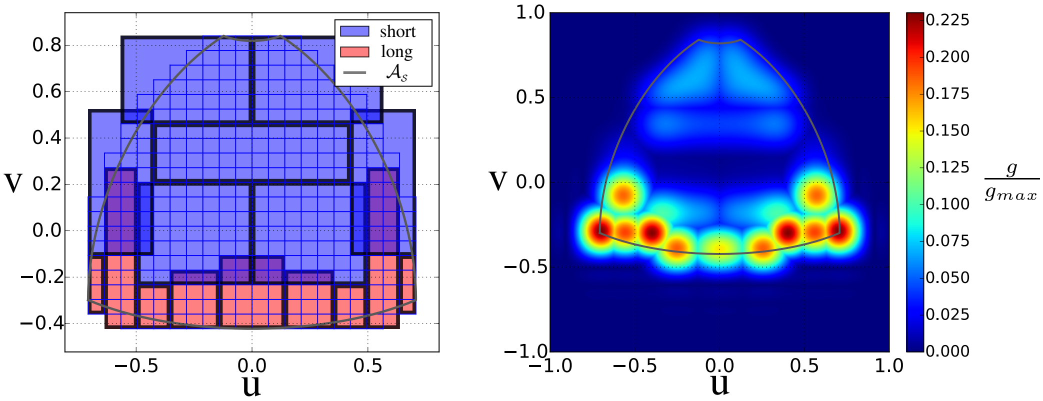

Figure 3.10: Radar search pattern obtained by branch-and-bound with long waveform in red and short waveform in blue (left), and total emission pattern (right)

emission gain is higher far from the antenna array normal direction, in order to compensate scanned losses. The detection range, displayed in Figure 3.11, shows that the radar pattern is over-energetic at high elevation. This can be explained by the reception digital beamforming processing constraint, which limits the area scanned by one dwell.

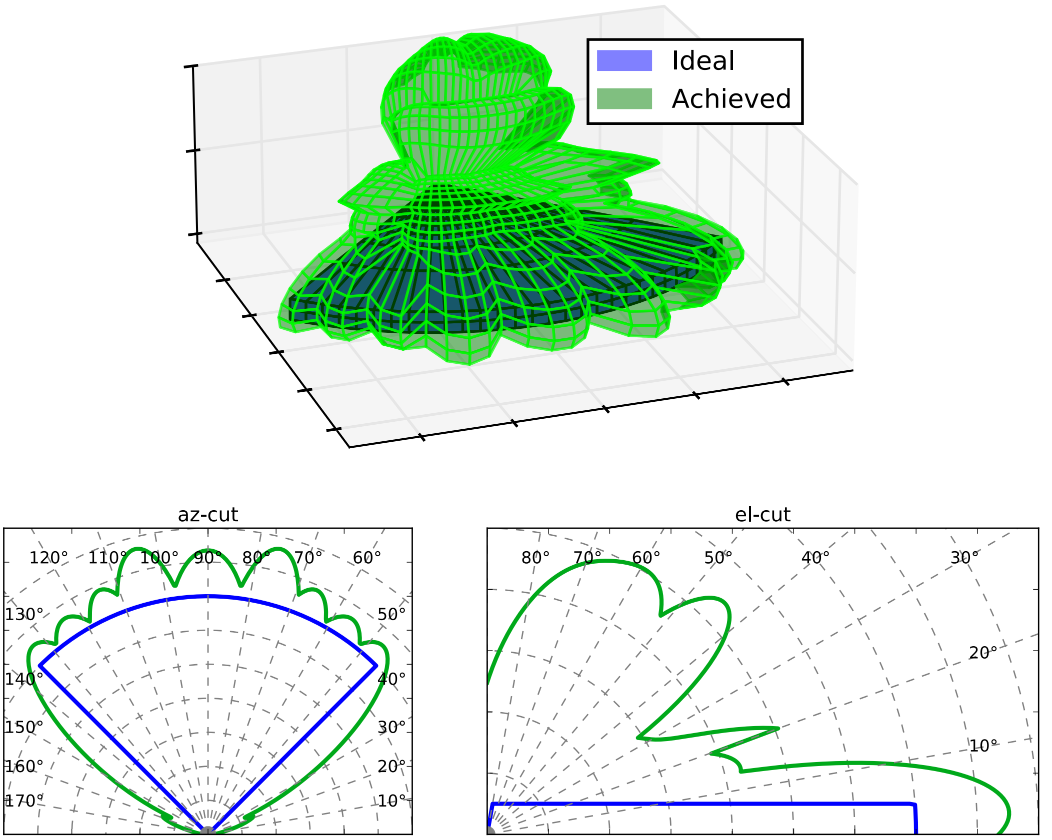

Figure 3.11: Detection range achieved by the solution in 3D (top), azimuth cut (bottom-left) and elevation cut (bottom-right)

Chapter 4

Extended formulations and computational improvements

The approximation of radar search pattern optimization as a set cover problem and its integer program formulation has various advantages. Integer programs are flexible tools, and can be extended to more powerful formulations of set covering, which can represent more complex problems in radar covering and account for additional operational requirements. Certain constraints, such as localized clutter and multiple missions can be integrated into the approximation model itself, with virtual no changes in the combinatorial cover problem structure. Other constraints, to be represented, need more general set covering formulations:

- *Set multicovering*: the problem where universe elements must be covered multiple times, which can represent scan update rate constraints in the context of radar optimization.

- *Probabilistic covering*: the problem where covers represent detection probabilities over the universe rather than its subsets. In the context of radar optimization, this approach can exploits dwell overlays and combine sub-energetic dwells to ensure global detection probability constraints.

As a major problem of combinatorial optimization, computational and practical aspects of the set cover problem have also been extensively studied [19, 46, 47]. Efficient, general-purpose integer programming solvers have been implemented and improved over the last decades [48, 18]. Those improvements offer now possibilities for research of multiple solutions [49, 50], and representation of the structure of the optimal set of a given problem, the set of optimal solutions to said problem.

On the other hand, the geometric characteristics of certain radar cover problems presented in this thesis can be exploited to implement efficient reduction methods. Those methods can reduce the number of variables and constraints in the problem, improving computational optimization but also representation of multiple solutions.

4.1 Additional constraints in radar operational optimization

Modern warfare requires from multi-function radars to ensures multiple tasks in complex situations [33].

In operational situation, the radar environment may not be uniform, and certain regions might have different properties in terms of clutter and terrain masking. Furthermore, the radar may receive informations from collaborating agents about incoming targets of interest. In such case, the radar could be required to increase its scan update rate in the targets incoming directions. An advantage of using a discrete grid for quantifying the surveillance area is the capacity for specifying those properties and constraints locally to the grid.

The radar might also have to search not one type of target, but multiple types (missiles, planes, etc.). While multiple search missions can sometimes be "combined" into a single mission, this may not always be the case, in particular for very different target types. An advantage of integer programming is that those multiple missions can be integrated by defining one detection constraint for each grid cell and each mission. All while using the same dwells to cover the surveillance space, and taking into account that each dwell might perform differently for each mission.

4.1.1 Localized constraints

Having localized constraints requires additional local information about the radar environment, see Figure 1.8, where for each direction:

- $\alpha : \mathcal{A}_S \to [0, 1[$ is the clutter eclipse coefficient. It represents the ratio of eclipsed area on the range-Doppler map in a given direction.

- $\mu : \mathcal{A}_S \to \mathbb{R}^+$ is the terrain masking distance, i.e. the maximum detection range in a given direction before terrain masks block detection.

Furthermore, the radar can be required to perform:

- $S_c : \mathcal{A}_S \to \mathbb{N}$ a minimum number of detection dwells ensuring that a desired scan update rate in a given direction is achieved. The local scan update rate is the number of detection dwells in the direction of interest over the total duration of the radar search pattern.

Taking into accounts those new parameters, the radar search pattern ensures detection if for each direction $(az, el) \in \mathcal{A}_S$, it contains at least $S_{c,i}(az, el)$ dwells, each capable of detecting a target with radar cross-section σ at range $\min\{\mu(az, el), R_c(az, el)\}$ with at least detection probability P_d and at most false alarm probability P_{fa} in clutter eclipse coefficient $\alpha(az, el)$.

4.1.2 Clutter and terrain masking

Localized clutter and terrain masking can be directly integrated into the computation of the dwell detection range. Taking into account terrain masking computationally simply requires to replace the desired detection by the terrain mask distance range, see Figure 4.1, since the radar cannot detect past the mask:

$$R_c(az, el) \leftarrow \min\{\mu(az, el), R_c(az, el)\}$$

In the combinatorial problem, clutter must be defined per cell, and thus has to be quantified over the grid. In other words, the clutter $\boldsymbol{\alpha}(m, n)$ is local to and constant within the grid cell $G_{m,n} \in G$, but can vary between grid cells. Various quantification scheme can be defined, with some examples shown in Figure 4.2:

- *erosion*: a grid cell contains a given clutter if it covers the entire cell.
- *dominant*: a grid cell contains a given clutter if it covers more than half the area in the cell.
- *dilatation*: a grid cell contains a given clutter if it covers a part of the cell, no matter how small.

When computing the detection range in a given cell (m, n) using the procedure in 1.6, the clutter is taken into account by using the waveform model described in 1.5 to compute the waveform required signal-to-noise ratio

$$s_w(P_d, P_{fa}, \boldsymbol{\alpha}(m, n))$$

Clutter is integrated during the approximation procedure in 3.3.1 and is virtually transparent to the combinatorial formulation in 3.3.3. Branch-and-bound optimization is thus not impacted by clutter.

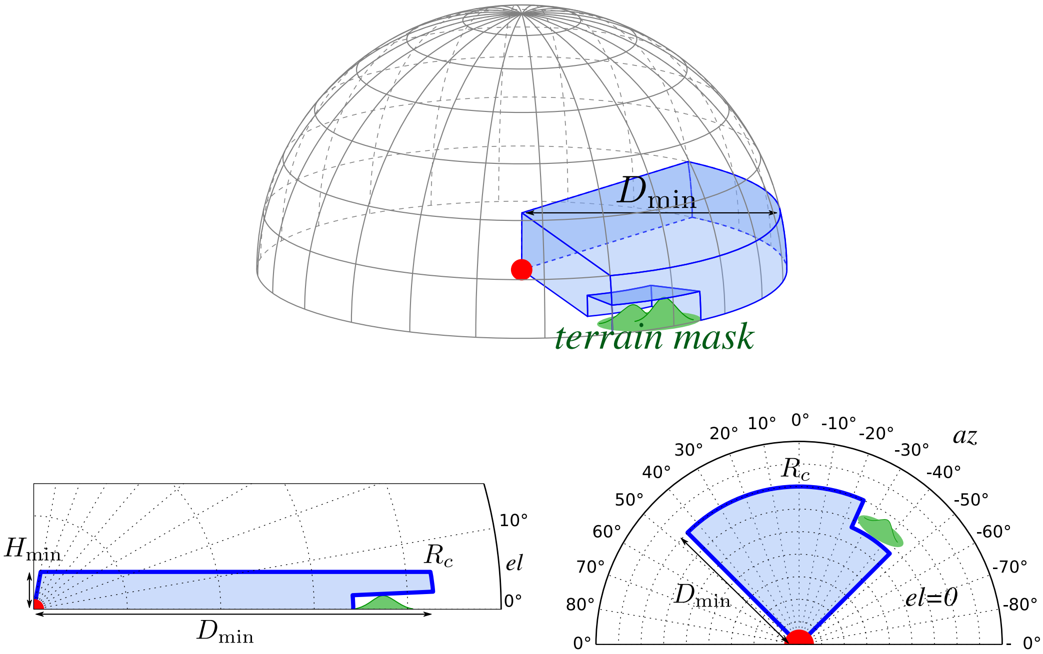

Figure 4.1: Masked desired detection range (top), azimuth cut (bottom-left), elevation cut (bottom-right)

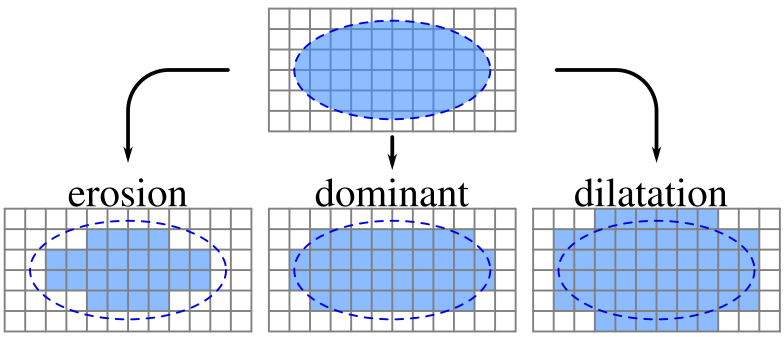

Figure 4.2: Quantification scheme for localized constraints

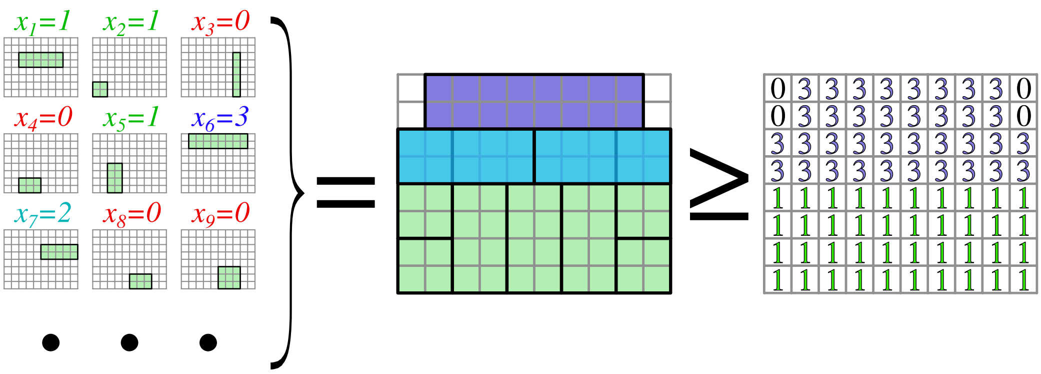

Figure 4.3: The set of available discrete covers with the chosen number of scan for each cover (left), the sum of the chosen discrete covers (middle) and the desired scan update rate for each cell (right)

4.1.3 Scan update rates

Similarly to clutter, the desired scan update rate must be quantified on the grid, using the same quantification schemes from Figure 4.2. The required number of scans $s(m,n)$ covering grid cell (m,n) is defined locally on the grid. But unlike local clutter, scan update rates constraints modify the combinatorial structure of the cover problem, as they requires an element to be covered multiple times, see Figure 4.3.

The generalized problem where the elements of the universe set must be covered multiple times is called the *set multicover problem*. The integer vector representation of the required number of scans is

$$\mathbf{s}(m+Mn) = s(m,n) \Leftrightarrow \mathbf{s} = \begin{pmatrix} s(0,0) \\ s(0,1) \\ \vdots \\ s(m,n) \\ \cdots \end{pmatrix}$$

and the corresponding integer program is

$$\begin{array}{rl} \min & \mathbf{T}^T.\mathbf{x} \\ s.t. & \mathbf{A}\cdot\mathbf{x} \geq \mathbf{s} \\ & \mathbf{x} \in \{0,\cdots,s_{\max}\}^p \subset \mathbb{N}^p \end{array} \tag{4.1}$$

with $s_{\max}$ be the maximum value of vector $\mathbf{s}$. The principal differences with integer program (2.2) are the right-handed side of the detection constraint being now the discrete required number of scans $\mathbf{s}$, and the variable vector $\mathbf{x}$

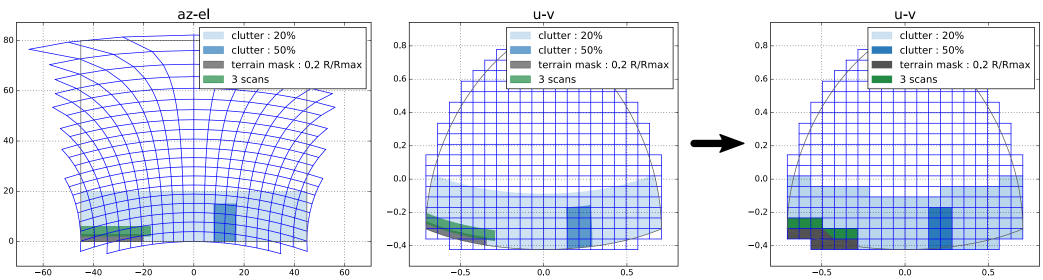

Figure 4.4: Mission constraints in azimuth-elevation (left), direction cosines (center) and discrete mission constraints (right)

now taking integer values. Branch-and-bound can by design optimize integer values with each branching representing the choice between multiple values, however with an increase in computational cost, because each node has $s_{\max}$ possible children.

Simulation example

The approximation procedure described previously was applied to a study case with 3 scan updates constraints, above a terrain mask, and with localized clutter. Constraints quantification followed the dominant scheme. Both original and quantified constraints are shown in Figure 4.4.

The PAR has 30×30 half-spaced radiating elements. The grid G is laid on a 20×20 lattice. We used a set with two possible waveforms $\mathcal{W} = \{w_1, w_2\}$, with a long waveform w_1 and a short waveform w_2. The approximation procedure produced 10943 feasible dwells. The detection grid contains 326 detection cells. The corresponding integer program has 10943 variables and 326 detection constraints.

The computation of the integer program is done in Python, and its optimization is done using CPLEX. The total time required to find the solution is 17 seconds on an i7-3770@3.4GHz processor with a random-access memory (RAM) usage of 420 megabytes. The solution, shown in Figures 4.5 and 4.6, uses 22 dwells to cover the surveillance area with 3 scan updates for certain dwells, but also combines slower scan update rates (1 or 2 updates) of overlapping dwells to achieve the desired global scan update rate.

4.1.4 Multiple missions model

In the case where the radar is tasked with multiple detection missions, its radar search pattern must ensure detection for a set of I missions. Parameters for each mission $i \in \mathcal{I} = \{1, \ldots, I\}$ are given:

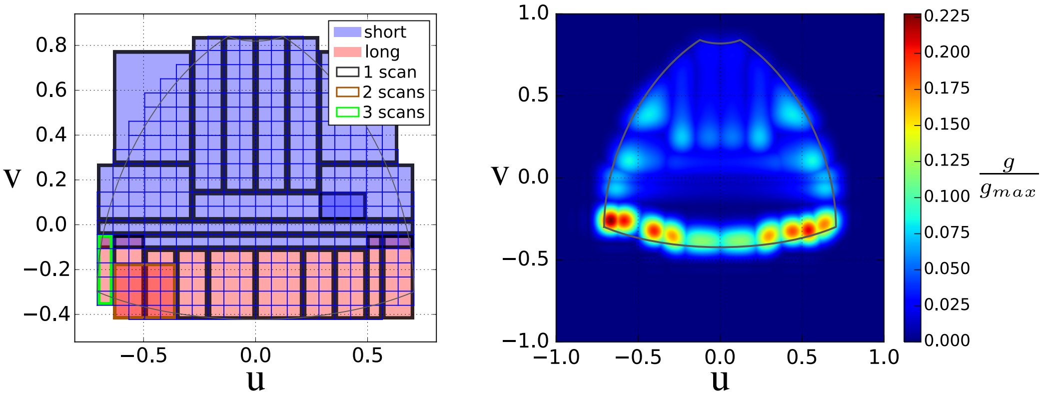

Figure 4.5: Radar search pattern obtained by branch-and-bound with long waveform in red and short waveform in blue (left), and total emission pattern (right)

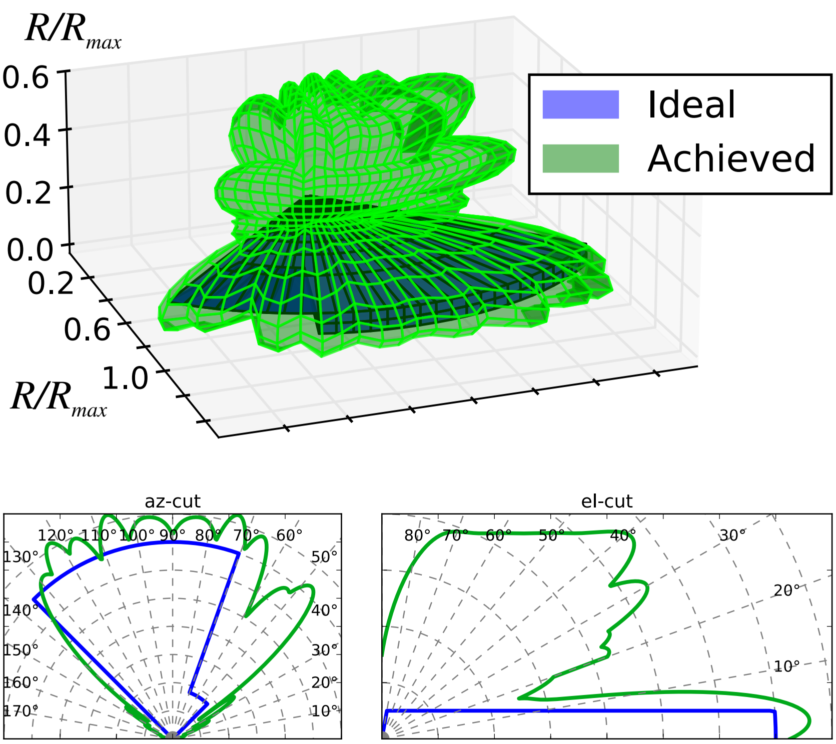

Figure 4.6: Detection range achieved by the solution in 3D (top), azimuth cut (bottom-left) and elevation cut (bottom-right)

- σ_i be the radar cross-section of the target type.
- $R_{c,i} : \mathcal{A}_S \to \mathbb{R}^+$ be the desired detection range.
- $S_{c,i} : \mathcal{A}_S \to \mathbb{N}$ be the desired scan update rate, which is the minimum number of scans to perform in a given direction during one radar search pattern.
- SW_i be the Swerling model [8].
- $P_d \in]0,1[$ is the desired detection probability and $P_{fa} \in]0,1[$ is the desired false alarm probability.

The radar search pattern ensures the required detection if for each mission i and each direction $(az, el) \in \mathcal{A}_S$, the radar search pattern contains at least $S_{c,i}(az, el)$ dwells, each capable of detecting a target with radar cross-section σ_i at range $\min\{\mu(az, el), R_{c,i}(az, el)\}$ with $\mu(az, el)$ the terrain masking range, with at least detection probability P_d and at most false alarm probability P_{fa} in clutter eclipse coefficient $\alpha(az, el)$.

Muli-mission pattern synthesis

Multiple missions have different energetic requirements. For each rectangle on the detection grid, the ideal radiation pattern for covering H for all mission at once is the maximum of each mission ideal radiation pattern is

$$g_H(u,v) \propto \begin{cases} L_s(u,v)^2 \max_i \left\{ \frac{R_{c,i}(u,v)^4 s_w(i,\alpha)}{\sigma_i} \right\} & \text{if } \begin{cases} u_n & \leq u & \leq u_q \\ v_m & \leq v & \leq v_r \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

up to a constant factor, as the array antenna feeds are normalized. Another possible approach is to consider a pattern for each rectangle and each mission.

Dwell discrete cover

For each dwell d_j in $\mathcal{D}$ and each mission i, the discrete cover $C_{j,i}$ of dwell d_j for mission i is computed through the same sampling methods presented in 3.3.1, using the dwell detection range $R_{j,i}$ and the mission desired detection range $R_{c,i}$. The discrete cover $C_{j,i}$ represents the cells on which dwell d_j validates mission i detection constraint.

So each dwell has multiple covers, one for each mission representing its detection performances on said mission, as shown in Figure 4.7 for two detection missions. A dwell cover can differ between missions, as each mission

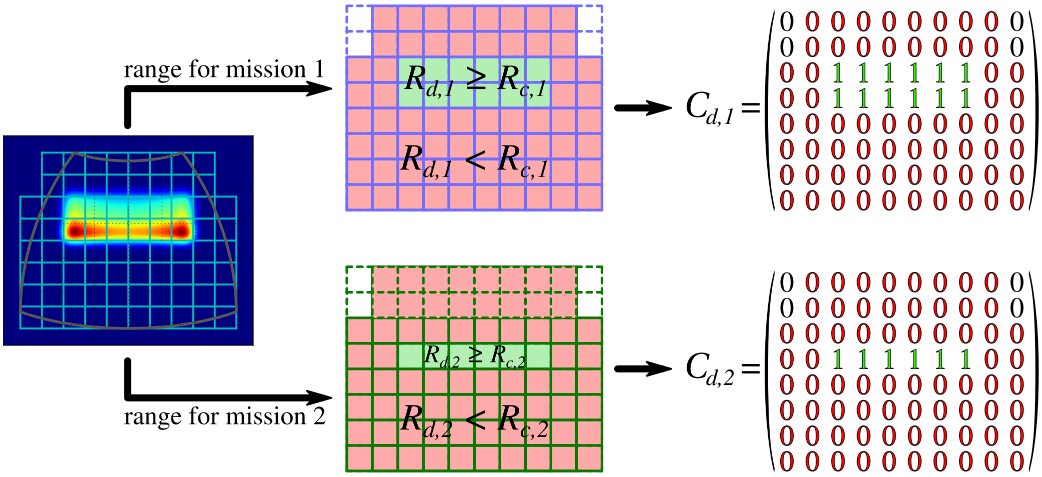

Figure 4.7: Computation of discrete covers for one dwell on two detection missions

has different energetic requirements and target type. Furthermore, some waveforms might be more efficient and suited for some missions.

From the combinatorial optimization perspective, each mission has a set of discrete covers of available discrete covers, and can be viewed as a set cover problem. Each mission $i \in \mathcal{I}$ has thus its own cover matrix and scan constraint vector such that minimization of the radar search pattern time-budget under detection constraints for all missions is

$$\begin{array}{rl} \min & \mathbf{T}^T.\mathbf{x} \\ s.t. & \forall i \in \mathcal{I}, \mathbf{A}_i \cdot \mathbf{x} \geq \mathbf{s}_i \\ & \mathbf{x} \in \{0, \cdots, s_{\max}\}^p \subset \mathbb{N}^p \end{array}$$

where $s_{\max}$ is the maximum value in all vectors $\mathbf{s}_i$. Each mission has different constraints but all missions use the same variables, and by combining all missions cover matrices in a unique matrix, and similarly all missions scan constraint vectors

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_i \\ \vdots \\ \mathbf{A}_I \end{pmatrix} \text{ and } \mathbf{s} = \begin{pmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_i \\ \vdots \\ \mathbf{s}_I \end{pmatrix}$$

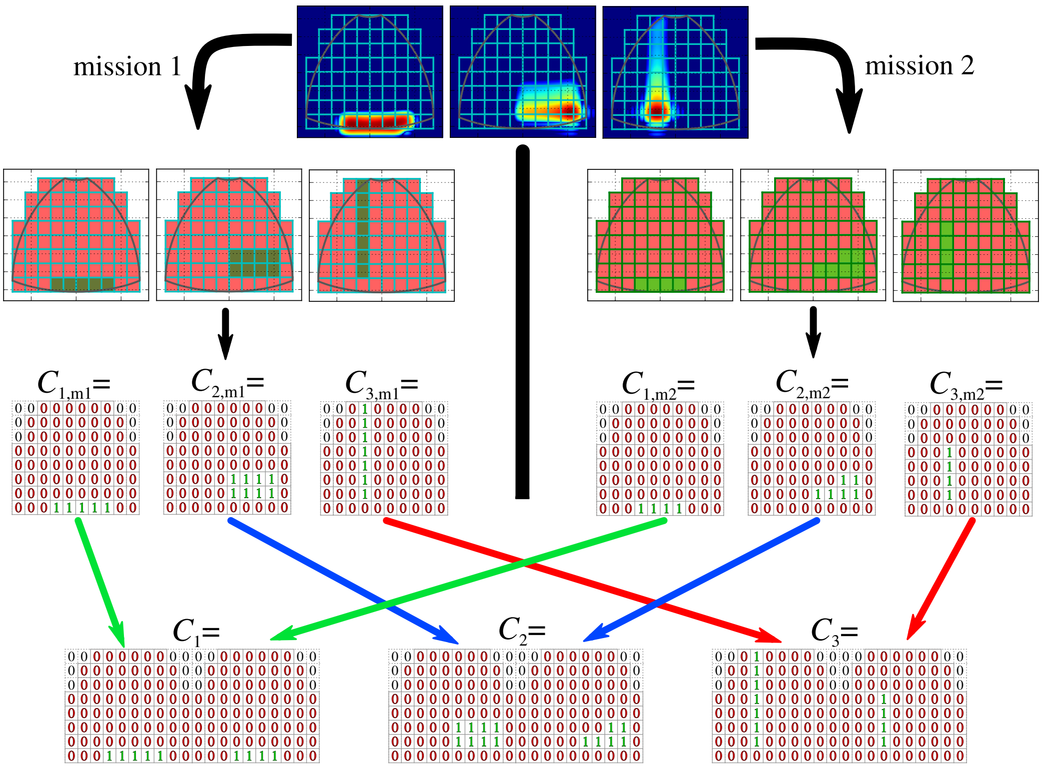

Figure 4.8: Two-missions combined covers

then the problem can be written as an integer program

$$
\begin{array}{rl}
\min & \mathbf{T}^T.\mathbf{x} \\
s.t. & \mathbf{A}\cdot\mathbf{x} \geq \mathbf{s} \\
 & \mathbf{x} \in \{0, \cdots, s_{\max}\}^p \subset \mathbb{N}^p
\end{array}
\tag{4.2}
$$

which virtually amounts to viewing each mission on a different grid and combining all those grids in one, as shown in Figure 4.8.

Simulation result

The multi-missions approximation procedure described above was applied to a study case with two scanning missions.

The radar array antenna has 30×30 half-spaced radiating elements. The grid G is laid on a 20×20 lattice. The radar has two available waveforms $\mathcal{W} = \{w_1, w_2\}$, with a long waveform w_1 and a short waveform w_2. The approximation procedure produced 30442 feasible dwells. The detection grid contains 326 cells for both scanning missions. The corresponding integer program has 30442 variables and 652 inequality constraints.

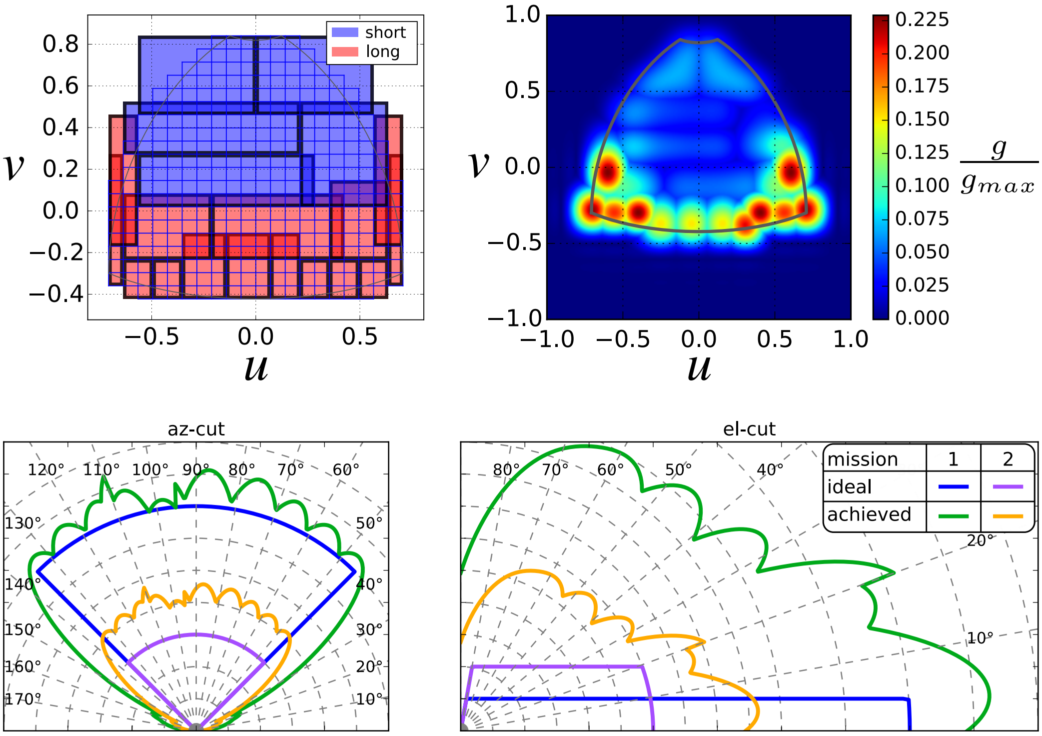

Figure 4.9: Radar search pattern obtained by branch-and-bound for two-missions case study

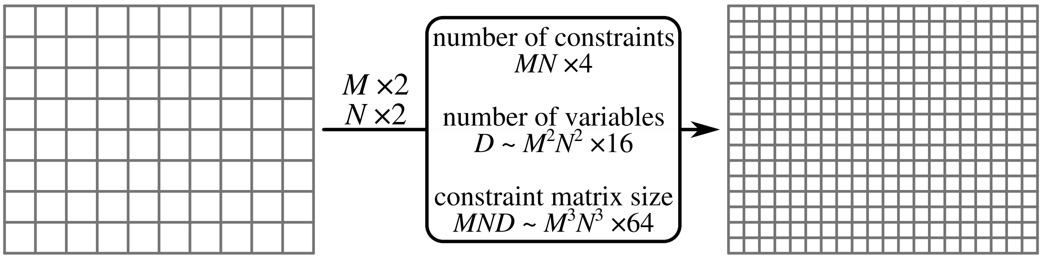

Figure 4.10: Effects of increasing the grid resolution by a 2-factor

The integer program is computed using Python, and optimization is done with CPLEX [45]. Total computation time for finding one optimal solution is 36 seconds on an i7-3770@3.4GHz processor with a memory usage of 450MB.

4.2 Pre-optimization reduction methods

The computational cost of optimization depends on the number of variables and constraints in the problem, especially for exploration methods such as branch-and-bound. The problem size is directly related to the detection grid size, i.e. the quantification resolution for the combinatorial problem. Increasing the resolution improves accuracy of the discrete approximation, and can improve the solution quality, but at the cost of increasing the number of variables and constraints, see Figure 4.10. In other words, having smaller detection cells and having more candidates dwells tends to improve the modelling but requires more computational time.

In rectangular grid covering, the number of constraints, or detection cells, is $O(MN)$. While the number of variables, or candidates dwells, increases in $O(M^2N^2)$, which is quadratically faster. The number of variables can quickly become a limiting factor for computational optimization of the radar search pattern.

A large number of those variables might not be required however. Certain dwell discrete covers might be redundant to each other. A cover is redundant if another cover can cover a wider area for an inferior cost. Reciprocally the latter cover is said to dominate the former. Formally, for $(C_a, C_b) \in \mathcal{C}^2$, C_a dominates C_b if:

- $\forall (m,n)\,, \mathbf{C}_a(m,n) \geq \mathbf{C}_b(m,n)$, i.e. C_a covers all cells covered by C_b.
- $T_a \leq T_b$, i.e. C_a costs less than C_b.

Transitivity, reflexivity and antisymmetry of domination are easily shown, and thus domination defines a partial order relation. Any solution using a

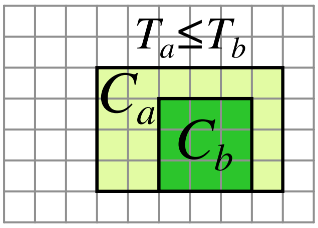

Figure 4.11: Cover domination of C_b by C_a

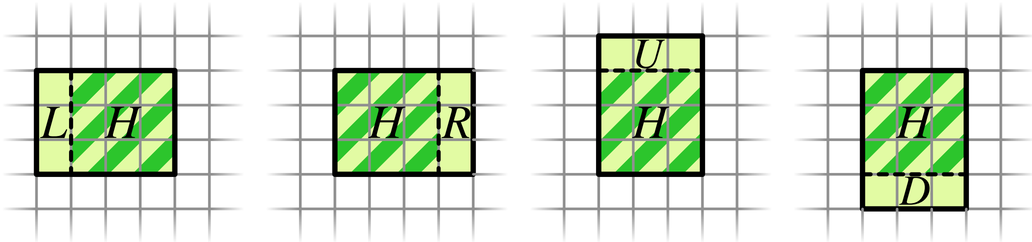

Figure 4.12: The four direct candidates rectangles L, R, U, D for finding a domination cover over rectangle H

dominated cover can be maintained or even improved by replacing the dominated cover by one of its dominating covers. Thus removing all dominated covers before branch-and-bound optimization does not change the optimal value of the problem instance, while diminishing the problem complexity.

4.2.1 Column reduction

Removal of dominated covers is equivalent to *column reduction*, a common technique in integer programming, often used before resolution to reduce the instance size [19, 51, 46]. The computational cost of a naive implementation for column reduction is $O(|\mathcal{C}|^2|G|)$. In rectangular grid covering for radar applications, where the number of candidates dwells grows with grid resolution in $O(|\mathcal{C}|) = O(M^2N^2)$, naive column reduction requires $O(M^5N^5)$ steps.

However, using the geometric characteristics of rectangular covers, column reduction can be performed in $O(M^2N^2)$ steps using $O(M^2N^2)$ space:

Loop through all possibles rectangles in decreasing size. For each rectangle H, check if it corresponds to an available cover C_a. Then check if any of the four rectangles obtained by increasing the width or height of H by 1, see Figure 4.12, can be covered by a cover C_b dominating C_a for a better cost. In that case, C_b covers H, and thus cover C_a can be removed from available covers. Algorithm 7 describes a pseudo-code of the procedure.

Column reduction "propagates" the domination relation among covers by

Algorithm 7 Column reduction

```
% Initialization and allocation of array of pointers to covers
Allocate an M × N × M × N pointer array p
for C ∈ 𝒞 do
    m, n ← coordinates of top-left corner of C
    h, w ← height and width of C
    Assign pointer p[m, n, h, w] to cover C
end for

% Loop through all possible rectangles by decreasing size
for (h, w) ∈ {M, ..., 1} × {N, ..., 1} do
    for (m, n) ∈ {0, ..., M − h} × {0, ..., N − w} do
        if p[m, n, h, w] is a cover then
            C_a ← p[m, n, h, w]

            % Get the dominating cover candidates, see Figure 4.12
            L ← p[m, n − 1, h, w + 1] (if it exists)
            R ← p[m, n, h, w + 1] (if it exists)
            U ← p[m − 1, n, h + 1, w] (if it exists)
            D ← p[m, n, h + 1, w] (if it exists)
            Get cover C_b with minimum cost among {L, R, U, D}

            % Update best cover for rectangle defined by [m,n,h,w]
            if T_a ≥ T_b then                                    ▷
                Delete C_a, assign pointer p[m, n, h, w] to cover C_b
            end if
        end if
    end for
end for
```

decreasing size, and ensure that all dominated covers are removed. Indeed, for any pair of covers (C_a, C_b) such that C_a dominates C_b, there is a sequence of rectangles from C_a to C_b, where each step of the sequence amounts to decreasing the height or width of the rectangle by 1, see Figure 4.13.

For each possible rectangle, the procedure search a minimum among 4 possibles values. Since there are $M(M+1)N(N+1)/4$ possible rectangles on grid G, Algorithm 7 requires $O(M^2N^2)$ steps. It also requires an array of size M^2N^2. However, only $M(M+1)N(N+1)/4$ entries in the array represent valid rectangles, so almost 75% of the array is not used. If memory usage is an issue, a more compact array can use instead the custom hash

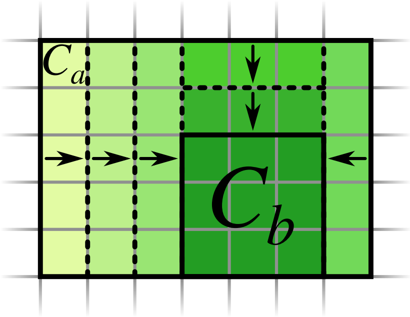

Figure 4.13: Sequence of dominating covers between two covers C_b and C_a

function

$$(m, n, h, w) \to \frac{(M-h+1)(M-h)(N+1)N}{4} + \frac{(M-h+1)(N-w+1)(N-w)}{2} + m(N-w+1) + n \quad (4.3)$$

which maps each subrectangle in G to a unique index in $[0, M(M+1)N(N+1)/4[$, see Appendix A for more details.

In radar search patterns, domination relation between covers is common and due to narrow over-energetic radiation patterns, which performs less efficient covering than widened radiation patterns. In numerical simulations, column reduction is rather efficient in decreasing the number of variables in the integer program.

4.2.2 Row reduction

Another common method for decreasing the instance size of integer program is *row reduction*, which removes redundant constraints. In the context of covering problems, a cell is redundant respectfully to another cell if the detection constraint of the former is necessarily validated by the detection constraint of the latter, see Figure 4.14

Formally, $\forall(g_a, g_b) \in G^2$, g_b is redundant in respect to g_a if and only if $\forall C \in \mathcal{C}, \mathbf{C}(m_b, n_b) \geq \mathbf{C}(m_a, m_a)$, where (m_a, n_a) are the coordinates of cell g_a and (m_b, n_b) the coordinates of g_b. Thus any cover including g_a also cover g_b. Reciprocally, g_a is said to imply g_b.

Removing redundant cells does not impact the optimal value of the problem instance. Similarly than for column reduction, naive row reduction requires $O(|G|^2|C|) = O(M^2N^2|C|)$, but can be reduced to $O(MN|C|)$ exploiting the geometrical properties of rectangular covers.

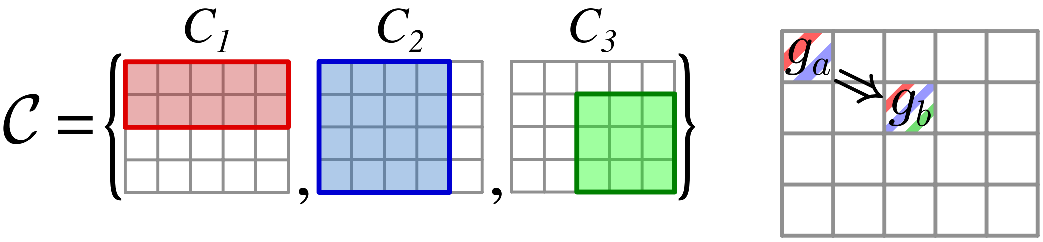

Figure 4.14: Cell g_a implies g_b, or reciprocally g_b is redundant to g_a (right) for given problem instance (left)

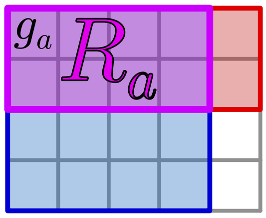

Figure 4.15: Rectangle R_a as the intersection of all covers including g_a (right) for problem instance of Figure 4.14

Let g_a be a cell. Let R_a be the intersection of all covers of $\mathcal{C}$ which includes g_a, see Figure 4.15:

$$R_a = \bigcap_{C \in \mathcal{C} : g_a \in C} C$$

and R_a is an intersection of parallel rectangles, and is not the null set since it contains at least g_a, so R_a is rectangle itself. The top-left (bottom-right) corner of R_a can be computed by taking the maximum (minimum) coordinates among top-left (bottom-right) corners of covers in $\{C \in \mathcal{C} : g_a \in C\}$.

It is straightforward from the definition of redundancy that all cells in R_a are redundant to g_a, since any cover including g_a also covers R_a. This property remains true in the general set cover problem. The advantage with rectangular covers is that intersections of rectangles are much easier to compute by using the convexity of rectangles.

By transitivity of redundancy, Algorithm 8 always keep for each removed cell at least one cell which implies the removed cell, directly or indirectly. On the other hand, for any pair of cells g_a, g_b such that g_a implies g_b, all covers including g_a covers g_b, thus $g_b \in R_a$, and R_a contains the rectangle formed using g_a and g_b as corners. So all redundant cells are removed.

Each cell is looped through at most twice, once in the main loop, and once when it is removed. For each cell, each cover is looped through twice, once to check if it contains the cell and once for computing intersection.The

Algorithm 8 Row reduction

```
% Loop through all cells
for g_a ∈ G do
    Allocate list of covers containing g_a: 𝒞_a = ∅
    for C ∈ 𝒞 do
        if g_a ∈ C then
            Add cover: 𝒞_a ← 𝒞_a ∪ {C}
        end if
        Compute intersection of covers: R_a ← ⋂_{C∈𝒞:g_a∈C} C
        for g_b ∈ R_a \ {g_a} do
            Remove redundant cells: G ← G \ {g_b}
        end for
    end for
end for
```

gain over the general set cover case is in computation of the intersection of covers, which takes $O(|C||G|)$ in general, but can performed in $4|C|$ steps with rectangular covers, with two maximum and two minimum searches of the corners of R_a.

Simulation results

The reduction gain for problems with various square grid size ($M = N$) is shown in Figure 4.16, where column reduction is shown to be highly effective in decreasing the number of variables and the memory usage, almost by a factor 10. Row reduction, while still relatively efficient in reducing the number of constraints, intrinsically operates on a smaller number of constraints, and has a negligible impact on memory performances.

4.3 Multiple-solution generation and representation

4.3.1 Branch-and-bound enumeration

While the branch-and-bound exploration could terminate once an optimal (or sufficiently near-optimal) solution is found, it is possible to expand and pursue the exploration of the search tree in order to enumerate alternative optimal solutions [49], but there is a trade-off between the computational/memory cost and exhaustiveness of the enumeration.

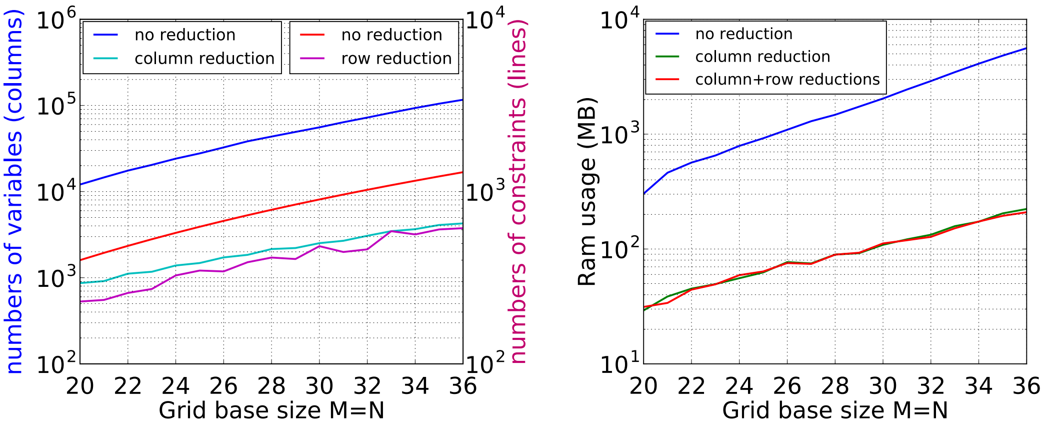

Figure 4.16: Number of columns and rows (left) and RAM usage (right) depending on reduction method(s) used.

In radar design and operational use, multiple solutions are a desirable feature. Search patterns have been hand-designed by engineers for decades, who have a strong expertise on the subject and prefer to use optimization as an aid-design tool. Similarly, radar operators preferred to have choice and flexibility between multiple modes in operational situations. Criterion such as bandwidth occupation, range resolution, system overheat, etc. can vary between different solutions, and their importance is usually dependant on the radar system characteristics and on the operational situation.

This choice in turn can be analysed to define preferences, to add secondary selection criterion to the method or even refined the model into a multi-objective optimization problem.

Multiple solutions enumeration can be done by slightly modifying steps **2.** and **3.** of the branch-and-bound method:

2. Bounding:
If the current node relaxation value is less than or equal to the current best solution found, proceed to Step 3, otherwise, drop current node and go back to Step 1.

3. Update and Enumerate:
If the current node relaxation is an integral solution, then its an improving solution. If it is strictly better than the current solution, empty the set of best solutions and update best current solution. Otherwise, update the set of best solutions. Proceed to Step 4 (as there could be other optimal solutions among the children of the current node).

This result in modifications to Algorithm 5 pseudo-code as described in Algorithm 9.

Algorithm 9 Branch-and-bound enumeration

```
% Initialization
...
x_best := x_F = (1···1)        ▷ Best solution found so far (by default, x_F is a valid solution)
𝒳_best := {x_F}                ▷ Set of best solutions found so far

% Exploration
while 𝒩 is not empty do
    ...

    % Bounding
    if T^T · x_L ≤ T^T · x_best then   ▷ Explore N if its relaxation is at least as good as x_best

        % Update and Enumerate
        if x_L ∈ {0,1}^D then          ▷ Check if x_L is an integral solution
            if T^T · x_L < T^T · x_best then
                x_best := x_L
                𝒳_best := {x_L}
            else
                𝒳_best := 𝒳_best ∪ {x_L}
            end if
        end if

        % Branching
        for x ∈ {0,1} do
            ...
        end for
    end if
end while
return 𝒳_best
```

4.3.2 Example

The branch-and-bound enumeration applied to the example given in 2.6.3 would keep searching after finding the solution, and would follow the steps

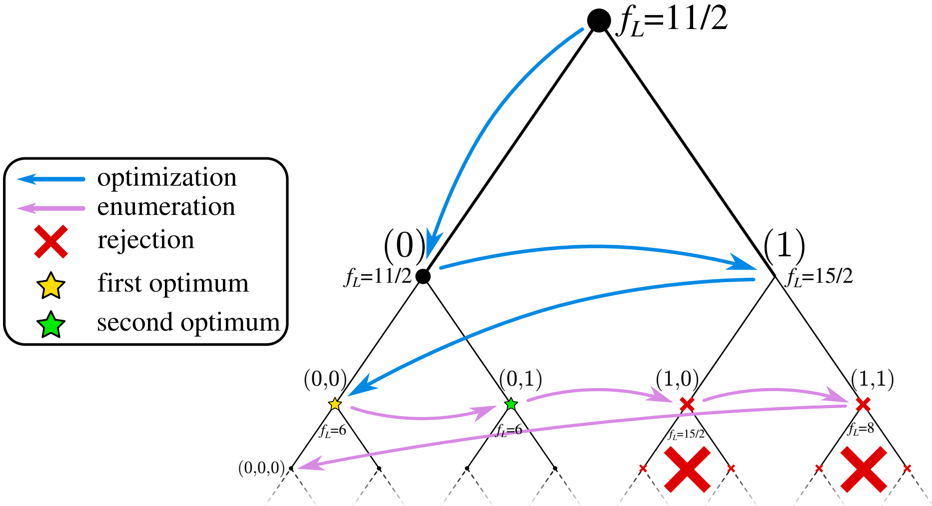

Figure 4.17: Enumeration branch-and-bound application example

below, see Figure 4.17:

- $\mathcal{N} = \{(0,0),(0,1),(1,0),(1,1)\}$, $\mathbf{x}_{best} = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$, $f_{best} = \mathbf{T}^T \cdot \mathbf{x}_{best} = 13$:
  Relaxation of $(0,0)$ yields the linear optimal solution $\mathbf{x}_L = (0\,0\,1\,1\,0\,0\,1\,1)$ with cost $6 \leq 13$. $\mathbf{x}_L$ is an integral solution, thus we update the best current solution $\mathbf{x}_{best} := \mathbf{x}_L$; $f_{best} := 6$.
  We add the children $(0,0,0)$ and $(0,0,1)$ to the exploration list $\mathcal{N}$.

- $\mathcal{N} = \{(0,1),(1,0),(1,1),(0,0,0),(0,0,1)\}$, $\mathbf{x}_{best} = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)$, $f_{best} = \mathbf{T}^T \cdot \mathbf{x}_{best} = 6$:
  Relaxation of $(0,1)$ yields the linear optimal solution $\mathbf{x}_1 = (0\,1\,1\,0\,0\,1\,1\,0)$ with cost $6 \leq 6$. $\mathbf{x}_1$ is an integral solution, thus added to $\mathcal{X}_{best} := \{\mathbf{x}_{best}, \mathbf{x}_1\}$. We add the children $(0,1,0)$ and $(0,1,1)$ to the exploration list $\mathcal{N}$.

- $\mathcal{N} = \{(1,0),(1,1),(0,0,0),\ldots\}$, $\mathbf{x}_{best} = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)$, $f_{best} = \mathbf{T}^T \cdot \mathbf{x}_{best} = 6$:
  Relaxation of $(1,0)$ yields the linear optimal solution $\mathbf{x}_L = (1\,0\,1\,1\,\frac{1}{2}\,\frac{1}{2}\,\frac{1}{2}\,\frac{1}{2})$ with cost $\frac{15}{2} > 6$. We drop node $(1,0)$ and proceed with the next node.

- $\mathcal{N} = \{(1,1),(0,0,0),\ldots\}$, $\mathbf{x}_{best} = (0\,0\,1\,1\,0\,0\,1\,1)$, $f_{best} = \mathbf{T}^T \cdot \mathbf{x}_{best} = 6$:
  Relaxation of $(1,1)$ yields the linear optimal solution $\mathbf{x}_L = (1\,1\,1\,0\,0\,0\,1\,1)$ with cost $8 > 6$. We drop node $(1,1)$ and proceed with the next node.

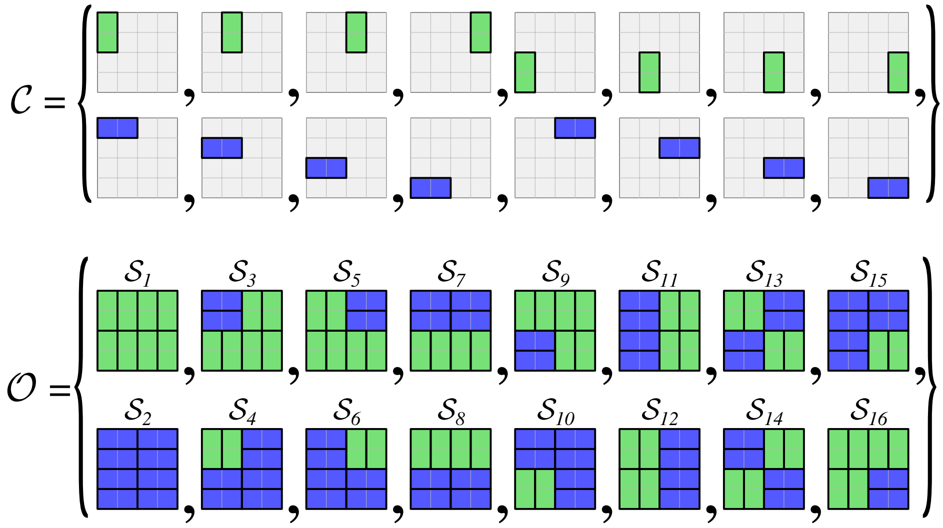

Figure 4.18: Collection of available covers (top), set of optimal solutions for the associated cover problem (bottom)

4.3.3 Exhaustive enumeration redundancy

In radar applications practical cases, there is usually a broad choice of possibility for cover problems, and therefore a large number of possible good-quality solutions. However, straightforward branch-and-bound enumeration can produce a lot of redundancy among solutions.

Figure 4.18 displays a problem instance example for which there is many redundant optimal solutions. Whereas all discrete covers are used by the union of pattern $\mathcal{S}_1$ and pattern $\mathcal{S}_2$, making this pair of solutions enough to represent "all covering possibilities". There are however 14 supplementary possible optimal solutions, which can be viewed as recombinations of covers in $\mathcal{S}_1 \cup \mathcal{S}_2$. These solutions bring no new information on which covers can be used to produce a new solution pattern, and many have significantly similar structure up to a vertical or horizontal symmetry. This phenomenon is caused by the presence of optimal sub-structures in the covering, i.e. different ways to cover the same area. The number of possible optimal solutions grows exponentially with the number of alternatives sub-structures. In Figure 4.18, there are four 4-by-4 sub-structures, one in each corner; and each has 2 alternatives optimal covering, horizontal or vertical, resulting in the 16 possibles solutions.

Solution redundancy is recurring problem in multiple solution generation which has already been discussed in [49, 50, 52, 53], against which the most common solution is to use diversity measures, for example based on string distances such as the Hamming distance.

Another way to avoid redundancy is to search for solutions which are not recombinations of previously known solutions. This can be done by maximizing an innovation metric, which would measures how different a new solution compared to all known previous solutions.

4.3.4 Innovation metric

Having multiple optimal solution gives alternative ways to solve the covering problems, but it also gives information about which covers are used in optimal solutions, in other words, which covers can be used to construct an optimal solution. Let $\mathcal{O} = \{\mathbf{y} \in \{0,1\}^D\}$ be a set of known optimal solutions, the cover indicator of $\mathcal{O}$ can be defined as the vector $\mathbf{o} = (o_i)_{i \in [1,D]}$ with

$$o_i = \max_{\mathbf{y} \in \mathcal{O}} \{y_i\}$$

and thus $\mathbf{o} = \bigvee_{\mathbf{y} \in \mathcal{O}} \mathbf{y}$ where $\bigvee$ is the logical bitwise OR operator applied to all solution vectors $\mathbf{y}$ as if they were bit vectors. The cover indicator represents the covers used in at least one solution of $\mathcal{O}$. Finding new optimal solutions, which use different covers compared to known solutions, will brings diversity to the set of solutions. More importantly it will increases the number of covers which can be used to construct optimal solutions. The number of "new covers" used by a solution $\mathbf{x}$ is measured by the innovation metric of $\mathcal{O}$

$$d(\mathbf{x}, \mathcal{O}) = \sum_{i=1}^{D} x_i(1 - o_i) = (\mathbf{1} - \mathbf{o})^T \cdot \mathbf{x} = \mathbf{d}^T \cdot \mathbf{x}$$

where $\mathbf{d} = (\mathbf{1} - \mathbf{o})$ is the cost vector of the metric. The metric can thus be written as a linear cost function. Informally, this metric counts how many covers used in solution $\mathbf{x}$ are not used by any solution $\mathbf{y} \in \mathcal{O}$. Diversity string-based metrics have already been used in generation of multiple solution in The difference with previous Hamming-like metrics is that the innovation metric does not penalize re-use of covers already used by solutions in $\mathcal{O}$. It only quantifies how many "not-previously-used" covers the new solution brings in $\mathcal{O}$. By extension, any discrete cover used in at least one optimal solution is defined as an "optimal candidate cover".

4.3.5 Innovation maximization problem

Sequential optimization is a common approach for generate multiple solutions [49]. The original problem is first solved, returning a first solution, from which the optimal cost value can be computed. The original cost function can then be reformulated as an equality constraint. This opens the possibility to use another metric as the cost function, like a diversity distance, or the innovation metric described above.

Conceptually, generating multiple solutions is no longer a minimization problem, as there is no need to search the optimal value since it is known. Whereas maximizing the innovation metric will produce more information on alternative ways to solve the problem. Since the innovation metric is a linear function, the maximization problem for finding a new solution $\mathbf{x}$ is an integer program

$$\begin{array}{rl} \max & \mathbf{d}^T.\mathbf{x} \\ s.t. & \mathbf{A}\cdot\mathbf{x} \geq \mathbf{1} \\ & \mathbf{T}^T.\mathbf{x} = f_{opt} \\ & \mathbf{x} \in \{0,1\}^D \end{array} \tag{4.4}$$

where $\mathcal{O}$ is the set of known previous optimal solution, and f_{opt} is the optimal cost value.

4.3.6 Iterative enumeration

After solving the original problem once, multiple new solutions can be generated by solving sequential instances of integer programs (4.4). At each iteration, the innovation metric is updated with the information received from the new solution. Algorithm 10 details the iterative enumeration, where $\bigwedge$ represents the logical bitwise AND operator.

Algorithm 10 has two useful features. Let $\mathbf{d}_k$ be the value of innovation cost vector $\mathbf{d}$ during the k-th step of the while loop:

- **Monotony**: by optimality of the k-th maximization problem solution

  $$\mathbf{d}_k{}^T \cdot \mathbf{x}_k \geq \mathbf{d}_k{}^T \cdot \mathbf{x}_{k+1}$$

  while each step $\mathbf{d} \leftarrow \mathbf{d}\bigwedge(\mathbf{1}-\mathbf{x}_i)$ in the algorithm "removes 1s from $\mathbf{d}$ and turn them in 0s", so $\{i : \mathbf{d}_{k+1}(i) = 1\} \subset \{i : \mathbf{d}_k(i) = 1\}$ which implies

  $$\forall \mathbf{x} \in \{0,1\}^D,\ \mathbf{d}_k{}^T \cdot \mathbf{x} = \sum_{i:\mathbf{d}_k(i)=1} x_i \geq \sum_{i:\mathbf{d}_{k+1}(i)=1} x_i = \mathbf{d}_{k+1}{}^T \cdot \mathbf{x}$$

  and combining both inequalities yields

  $$\mathbf{d}_k{}^T \cdot \mathbf{x}_k \geq \mathbf{d}_{k+1}{}^T \cdot \mathbf{x}_{k+1}$$

Algorithm 10 Iterative enumeration

% Solve the original problem and initialize parameters
$\mathbf{x}_0 \leftarrow \text{argmin}\{\mathbf{T}^T \cdot \mathbf{x} : \mathbf{A} \cdot \mathbf{x} \geq \mathbf{1} \wedge \mathbf{x} \in \{0,1\}^D\}$
$\mathcal{O} \leftarrow \{\mathbf{x}_0\}$
$d_{max} \leftarrow +\infty$
$\mathbf{d} \leftarrow \mathbf{1} - \mathbf{x}_0$

% Keep searching new solutions as long as they use yet-unused covers
while $d_{max} > 0$ **do**
$\quad\mathbf{x}_k \leftarrow \text{argmax}\{\mathbf{d}^T \cdot \mathbf{x} : \mathbf{A} \cdot \mathbf{x} \geq \mathbf{1} \wedge \mathbf{T}^T \cdot \mathbf{x} = f_{opt} \wedge \mathbf{x} \in \{0,1\}^D\}$

$\quad$% Update parameters
$\quad d_{max} \leftarrow \mathbf{d}^T \cdot \mathbf{x}_k$
$\quad\mathcal{O} \leftarrow \mathcal{O} \cup \{\mathbf{x}_k\}$
$\quad\mathbf{d} \leftarrow \mathbf{d} \bigwedge (\mathbf{1} - \mathbf{x}_k)$
end while

which means that the value of $d_{max} = \mathbf{d}_k \cdot \mathbf{x}_k$ decreases (usually non-strictly) when k increases. So the most different solution from previously known solutions are computed at the beginning of the loop. More importantly, at any step the value $d_{max} = \mathbf{d}_k \cdot \mathbf{x}_k$ indicates how many new covers each additional step can add at most.

- **Linearly bounded termination**: if an iteration returns a null maximum innovation $\mathbf{d}^T \cdot \mathbf{x}_k = 0$, then by optimality there is no "yet-unused" optimal cover left to find. By monotony, the sequence $(\mathbf{d}_l \cdot \mathbf{x}_l)_{l \geq k}$ is null for all subsequent searches anyway. Thus any optimal solution of integer program (2.2) will only use variables in the cover indicator $\mathbf{o} = \mathbf{1} - \mathbf{d}$. "$\mathbf{d}^T \cdot \mathbf{x}_k = 0$" is an *enumeration certificate*, which guarantees that any optimal solution can be constructed from known solutions.

  Furthermore, at each step where $\mathbf{d}_k \cdot \mathbf{x}_k > 0$, at least one new "yet-unused" optimal cover is found, so necessarily $\mathbf{d}$ has "at least a 1 removed", and since $\mathbf{d}$ is of length D, the while loop cannot perform more than D steps. The number of steps in Algorithm 10 is bounded by the number of variables, whereas a generic sequential algorithm for generating different solution may have an exponential number of steps, as some problem instances can yield an exponential number of different optimal solutions.

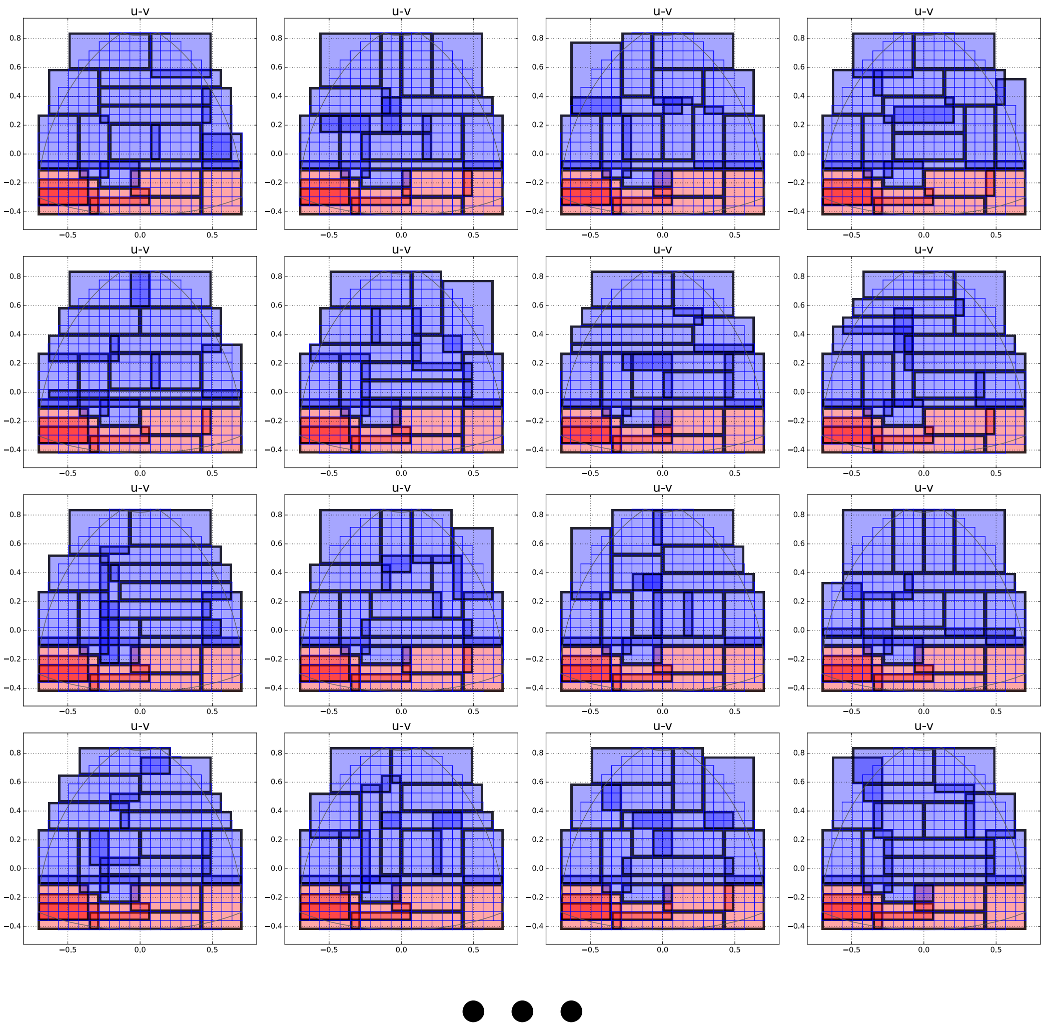

Figure 4.19: Multiple optimal solutions found by iterative enumeration

4.3.7 Optimal set structure

Using iterative enumeration provides multiple different solutions, see Figure 4.19, while ensuring solution diversity by maximizing a metric distance between solutions. However, Algorithm 10 main advantage is the computation of the *complete optimal cover indicator* $\overline{\mathbf{o}} = \mathbf{1} - \mathbf{d}$, containing all covers which can be used to produce an optimal solution. Whereas the set of all possible optimal solutions $\overline{\mathcal{O}}$ is usually too big to be computed in practice, the complete optimal cover indicator $\overline{\mathbf{o}}$ can still be used to analyse and exploit the structure of $\overline{\mathcal{O}}$.

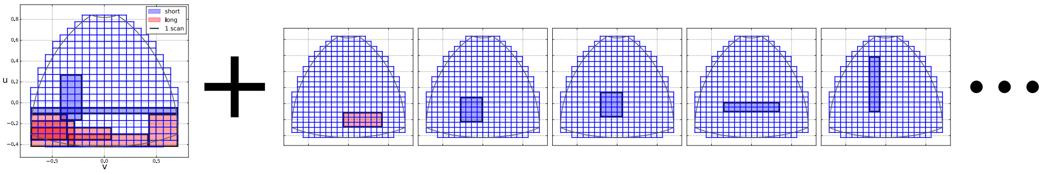

Figure 4.20: Any optimal solution is a combination of th optimality invariant (left) and selection of optional optimal covers (left)

Optimal column reduction

Knowing which covers are used in at least one optimal solution also implies by complementarity knowing which covers are not used by any optimal solution. Removing those covers from the set of available covers does not impact the set of optimal solutions: let $\mathcal{C}_{\overline{\mathbf{o}}} = \{C_i \in \mathcal{C} : \overline{o}_i = 1\}$. The reduced problem obtained by replacing $\mathcal{C} \leftarrow \mathcal{C}_{\overline{\mathbf{o}}}$ yields the same set of optimal solutions, as any optimal solution to the original problem is a solution to the reduced problem and vice-versa.

In fact, $\mathcal{C} \leftarrow \mathcal{C}_{\overline{\mathbf{o}}}$ corresponds to the optimal column reduction of the problem, i.e. the smallest subset of columns that preserves the set of optimal solutions to the problem.

Optimality invariant

An obvious reduction for covering problems is the case where one constraint can only be satisfied by one variable, in other words, a detection cell which can be covered by only one dwell. Similarly, if a constraint has a unique cover in the complete optimal cover indicator, then that cover is necessarily in any optimal solution. This cover is part of the *optimality invariant* of the problem, see Figure 4.20.

More generally, the optimality invariant $\mathring{\mathcal{I}}$ can be defined as the largest subset of covers which is contained in any optimal pattern

$$\mathring{\mathcal{I}} = \bigcap_{\mathbf{y} \in \overline{\mathcal{O}}} \{C_i : y_i = 1\}$$

an can be viewed as the "intersection" of all optimal solutions in $\overline{\mathcal{O}}$. Similarly to the complete optimal cover indicator $\overline{\mathbf{o}}$, the optimal invariant is represented by the *optimal invariant vector*

$$\bar{\mathbf{i}} = \bigwedge_{\mathbf{y} \in \overline{\mathcal{O}}} \mathbf{y}$$

The variables in the optimality invariant can be set as constants when constructing or modifying an optimal solution $\mathbf{x}$:

$$\forall j \in \{1, \ldots, D\},\ \mathring{i}_j = 1 \Rightarrow x_j = 1$$

as the covers in the optimality invariant cannot be replaced to produce a solution.

The concept of "invariant set" can be generalized to any set of optimal of solutions $\mathcal{O} \subset \overline{\mathcal{O}}$, for which the invariant set contains the covers who are part of all solutions in $\mathcal{O}$:

$$\mathcal{I} = \bigcap_{\mathbf{y} \in \mathcal{O}} \{C_i : y_i = 1\}$$

with its associated invariant vector being $\mathbf{i} = \bigwedge_{\mathbf{y} \in \mathcal{O}} \mathbf{y}$.

The optimality invariant can be viewed as the smallest invariant set

$$\mathring{\mathcal{I}} \subset \mathcal{I}$$

since $\mathcal{O}$ does not contain all optimal solutions, its invariant set $\mathcal{I}$ may contain cover which are not part of the optimality invariant, because an optimal solution not using them has not been found yet. As an example, for a set of optimal solution with only one solution, the invariant is the solution itself

$$\mathcal{O} = \{\mathbf{x}\} \Rightarrow \mathcal{I} = \{C_i : x_i = 1\} \Leftrightarrow \mathbf{i} = \mathbf{x}$$

as there is no information on other alternative solutions, and thus on which cover are obligatory, and which are not.

While the complete optimal set $\overline{\mathcal{O}}$ is not computable in general, computing the optimality invariant $\mathring{\mathcal{I}}$ can be done by iterative reduction of a known invariant set $\mathcal{I}$, where each step optimize an integer program

$$\begin{array}{rl} \min & \mathbf{i}^T.\mathbf{x} \\ s.t. & \mathbf{A} \cdot \mathbf{x} \geq \mathbf{1} \\ & \mathbf{T}^T.\mathbf{x} = f_{opt} \\ & \mathbf{x} \in \{0,1\}^D \end{array} \tag{4.5}$$

which search an optimal solution using the fewest possible number of covers from the current invariant. Iterative reduction is described in Algorithm 11. Note that if only one optimal solution $\mathbf{x}$ is known at initialization, the algorithm essentially starts with $\mathbf{i} \leftarrow \mathbf{x}$.

Using the same reasoning than for Algorithm 10, each step of iterative reduction "removes at least a 1 from $\mathbf{i}$". The stopping criteria r decreases

Algorithm 11 Iterative reduction for computing optimality invariant

% Start from a set of known set of optimal solution $\mathcal{O}$
$\mathbf{i} \leftarrow \bigwedge_{\mathbf{x} \in \mathcal{O}} \mathbf{x}$
$r \leftarrow 1$

% Search a solution not using all "candidate" invariant covers
while $r > 0$ **do**
  $\mathbf{x}_k \leftarrow \operatorname{argmin}\{\mathbf{i}^T \cdot \mathbf{x} : \mathbf{A} \cdot \mathbf{x} \geq \mathbf{1} \wedge \mathbf{T}^T \cdot \mathbf{x} = f_{opt} \wedge \mathbf{x} \in \{0,1\}^D\}$

  % Update parameters
  $r \leftarrow \mathbf{i}^T \cdot (\mathbf{1} - \mathbf{x}_k)$ % stopping criteria: number of removed covers in this iteration
  $\mathbf{i} \leftarrow \mathbf{i} \bigwedge \mathbf{x}_k$
end while

monotonously and reaches 0 in a number of steps bounded by, D, the number of candidates covers and the size of $\mathbf{i}$. When $r = 0$, then there is no optimal solution not using all covers in the "current" invariant.

In practice, even with few optimal solutions resulting from a premature stop of Algorithm 10, the initial invariant is the optimality invariant, and (4.5) is only solved once to ensure that there is no cover in the invariant unused by an optimal solution.

Choice metrics

The optimality invariant $\bar{\mathbf{i}}$ and the complete optimal cover indicator $\bar{\mathbf{o}}$ are the extreme descriptors of the optimal set structure

- $\bar{\mathbf{o}}$ describes the set of covers used in at least one optimal solution.
- $\bar{\mathbf{i}}$ describes the set of covers used in all optimal solutions.

The optimality invariant is the set of covers which cannot be replaced when modifying an optimal solution. This intuits the idea of hierarchy among covers, in terms how many alternatives there is for an optimal cover.

A straightforward generalization would be to count the number of solutions using a given cover. This criteria is however impractical, as it would requires to exhaustively enumerate all solutions, which infeasible in practice. However, it is possible to derive simpler metrics from the complete optimal indicator.

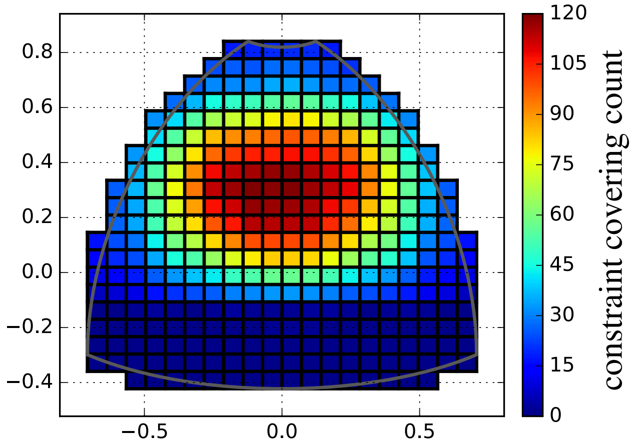

Figure 4.21: Constraint covering count: the number of optimal dwells covering each constraint

Constraint covering count

For each detection cell (i.e. constraint), the number of covers (i.e. variables) covering the cell give an indication of "how many alternate ways" to cover said cell exist:

$$\#g_{m,n} = |\{C \in \mathcal{C}_{\overline{\mathcal{O}}} : g_{m,n} \in C\}|$$

In practice, this classifies which cells gives less options in covering. Evidently, a cell with covering count of 1 has only one "possible choice", and the associated cover is part of the optimality invariant. Usually the grid side areas have a low count, contrary to the grid central area where more choices are available, see Figure 4.21.

Cover alternative count

From the previous metric, the *cover alternative count* of a given cover is the minimum value of covering count among covered constraints, and gives an indication of "how many alternatives" can replace the cover:

$$\#C = \min_{g \in C}\{\#g\}$$

A cover with an alternative count of 1 is in the optimality invariant, as there is a cell which can only be covered by this cover.

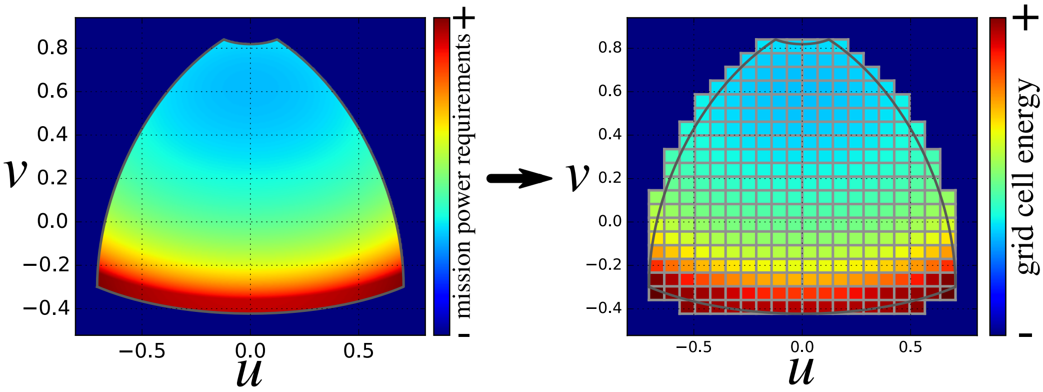

Figure 4.22: Continuous energy distribution e (left) and its quantification on the detection grid (right)

4.4 Future research leads

This section presents the theoretical work on two future research leads: grid adaptation, and probability covering for combining overlapping dwells.

4.4.1 Grid adaptation

Between the continuous general problem and its combinatorial approximation, quantification on the grid implies a lost of information. Optimal combinatorial solutions are possibly “sub-optimal” for the original continuous problem, and their accuracy likely depends on the grid resolution.

The grid resolution can be uniform, as has been done so far, such that every cell on the grid covers an equal area. Another possibility is to take irregular quantification step, with more precision in area more likely to require finer tuning of the search pattern.

Qualitatively, the total emitted power of a radiation pattern is constant, and the radar emits the same total power summing all directions. Spreading out the radiation pattern causes a proportionate decrease in angular power density. From an energetic point of view, radar covering can be viewed as using “energy shapes”, with each shape having the same total “energy”, to cover a space with energy requirements. This space is anisotropic though, and different areas requires different powers, see Figure 4.22.

Intuitively, a more adequate to quantify this space would be to somehow follow the energy requirements distribution, with smaller cells where requirements are higher, such that each cell represent the same energy requirement, see Figure 4.23.

Optimizing the grid to a certain energy density repartition $e : \mathcal{A}_S \to \mathbb{R}_+$

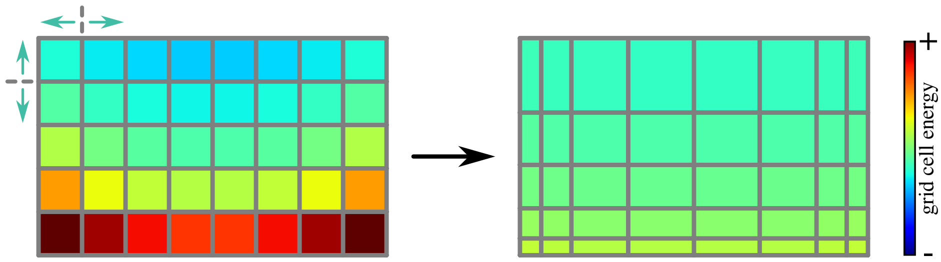

Figure 4.23: Optimization producing an irregular grid with a more even energy distribution

can be done in iterative manner. Starting from a given grid, the quantification values $\{u_0, \ldots, u_N\} \times \{v_0, \ldots, v_M\}$, corresponding to the grid nodes locations, are iteratively shifted, where at each step

- each value u_n, with $1 \leq n \leq N-1$ is shifted to the horizontal median $\hat{u}_n$ of its two surroundings columns which is the solution of

$$\int_{u_{n-1}}^{\hat{u}_n} \int_{v_0}^{v_M} e(u,v)\mathrm{d}u\mathrm{d}v = \frac{1}{2} \int_{u_{n-1}}^{u_{n+1}} \int_{v_0}^{v_M} e(u,v)\mathrm{d}u\mathrm{d}v$$

  which can be computed numerically by root-finding.

- each value v_m, with $1 \leq m \leq M-1$ is shifted to the vertical median $\hat{v}_m$ of its two surroundings rows and is which solution of

$$\int_{u_0}^{u_N} \int_{v_{m-1}}^{\hat{v}_m} e(u,v)\mathrm{d}u\mathrm{d}v = \frac{1}{2} \int_{u_0}^{u_N} \int_{v_{m-1}}^{v_{m+1}} e(u,v)\mathrm{d}u\mathrm{d}v$$

  which can be computed numerically by root-finding.

- the values u_0, u_N, v_0 and v_M remain unchanged, as those values defined the boundaries of the grid.

see Figure 4.24. The method requires numerical resolution of $N+M$ equations at each step, which might be computational costly. A more practical and conceptually close method is Lloyd's algorithm, also known as the Voronoi iteration, where at each step:

- each value u_n, $1 \leq n \leq N-1$ is shifted to the horizontal weighed centroid of its two surroundings columns

$$u_n \leftarrow \int_{u_{n-1}}^{u_{n+1}} \int_{v_0}^{v_M} ue(u,v)\mathrm{d}u\mathrm{d}v$$

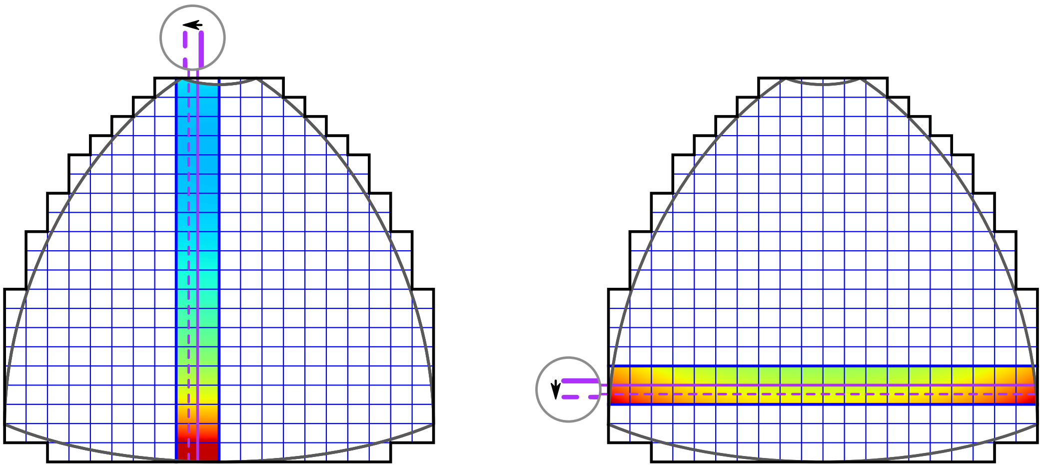

Figure 4.24: Computation of horizontal shift (left) and vertical shift (right) of a shift iteration

- each value v_m, $1 \leq m \leq M-1$ is shifted to the vertical weighed centroid of its two surroundings rows

$$v_m \leftarrow \int_{u_0}^{u_N} \int_{v_{m-1}}^{v_{m+1}} ve(u,v)\mathrm{d}u\mathrm{d}v$$

The two methods differs by the fact that the first method computes medians at each step, whereas the Voronoï iteration compute means.

The optimal quantification values $\{u_1^*, \ldots, u_{N-1}^*\}$ and $\{v_1^*, \ldots, v_{M-1}^*\}$ can also be found by numerically solving the system where the energy integral between each successive couple of values is equal:

$$\begin{cases} \forall n \in \{0, \ldots, N-1\}, & \int_{u_n}^{u_{n+1}} \int_{v_0}^{v_M} e(u,v)\mathrm{d}u\mathrm{d}v &= \frac{E_T}{N} \\ \forall m \in \{0, \ldots, M-1\}, & \int_{u_0}^{u_N} \int_{v_m}^{v_{m+1}} e(u,v)\mathrm{d}u\mathrm{d}v &= \frac{E_T}{M} \end{cases}$$

where $E_T = \int_{u_0}^{u_N} \int_{v_0}^{v_M} e(u,v)\mathrm{d}u\mathrm{d}v$ is the total energy requirement over the grid. This system can be solved by first computing E_T through numerical integration, then applying numerical root-finding to the series of functions

$$u_n \rightarrow \int_{u_0}^{u_n} \int_{v_0}^{v_M} e(u,v)\mathrm{d}u\mathrm{d}v - n\frac{E_T}{N}$$

to compute u_n. Similarly, numerical root-finding is used on the series of functions

$$v_m \rightarrow \int_{u_0}^{u_N} \int_{v_0}^{v_m} e(u,v)\mathrm{d}u\mathrm{d}v - m\frac{E_T}{M}$$

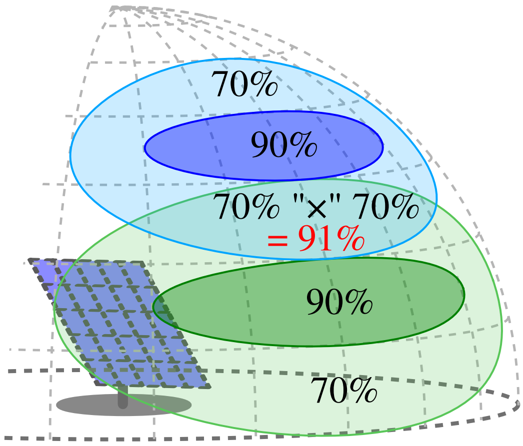

Figure 4.25: Overlay of two dwells and overall detection probability

to compute v_m.

Note that all those methods might produce a highly irregular grid, and thus may need to be constrained in practice, for example ensuring that a cell size cannot go below or above certain bound values. Those values could be derived from the radar narrow beam-width for the lower bound, and the radar maximum scanning area for the upper bound.

4.4.2 Probability covering

So far, in all presented formulations, the desired detection probability is achieved independently by each dwell. However, overlapping multiple dwells can improve the overall detection probability, the probability that at least one dwell achieve detection.

For example, if a target is covered by two dwells, each with 70% probability, the overall probability that at least one of the dwells detects the target is $1 - (1 - 70\%)^2 = 91\%$. For a desired detection probability of 90%, both dwells individually fail the requirement while the combination of both dwells achieves it globally, see Figure 4.25.

Within this approach, a dwell is no longer represented by a discrete combinatorial cover, encoded in “0s” and “1s”, but by a quantified probability cover, see Figure 4.26, representing the dwell detection probability inside each grid detection cell.

A cell (m, n) local detection probability $p_j(m, n)$ of a dwell $d_j \in \mathcal{D}$ can be computed by inverting the radar equation (1.10) into

$$s_w = \frac{P_p\ f_w\ T_w\ g_e\ g_r\ {\lambda_w}^2\ \sigma}{(4\pi)^3\ {R_c}^4\ L_u\ {L_s}^2} \tag{4.6}$$

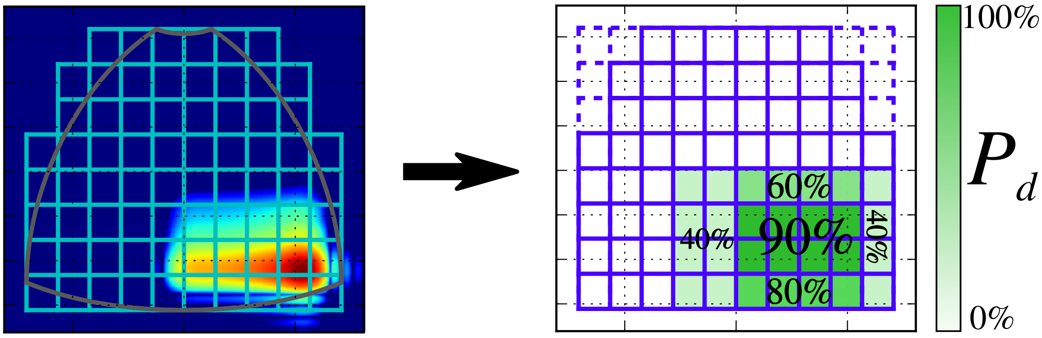

Figure 4.26: Overlay of two dwells and overall detection probability

with R_c the required detection range. From the signal-to-noise s_w ratio for a target echo at range R_c, the detection probability is either known for waveform measured performances, or can be computed using the waveform model from 1.5, and equations (1.7), (1.8) and (1.9).

The detection constraint is no longer to have one dwell ensuring detection, but to ensure an overall minimum detection probability P_D by combining multiple dwells. Equivalently, it can also be said the detection constraint ensures that the probability of global failed detection must be below the acceptable failure probability $1 - P_D$. If dwell detections are seen as independent events, which is true for detection tests polluted by white noise, then the global failure probability for a cell (m, n) is

$$\prod_{j=1}^{D}(1 - p_j(m,n)) \leq 1 - P_D(m,n)$$

which can be linearised by applying the logarithm function into

$$\sum_{j=1}^{D}\ln(1 - p_j(m,n)) \leq \ln(1 - P_D(m,n))$$

For cell (m, n), let

- $l_j(m,n) = \ln(1 - p_j(m,n))$ the anti-log dwell detection probability
- $b(m,n) = \ln(1 - P_D(m,n))$ the anti-log desired detection probability

and the detection constraint becomes the linear inequality

$$\sum_{j=1}^{D} l_j(m,n) \leq b(m,n)$$

Let the anti-log dwells detection probability matrix $\mathbf{L}$ and anti-log desired detection probability vector $\mathbf{b}$ be defined as

$$\mathbf{L} = \begin{pmatrix} l_1(0,0) & \cdots & l_D(0,0) \\ l_1(1,0) & \cdots & l_D(1,0) \\ \vdots & \ddots & \vdots \\ l_1(m,n) & \cdots & l_D(m,n) \\ \vdots & \vdots & \vdots \end{pmatrix}, \mathbf{b} = \begin{pmatrix} b(0,0) \\ b(0,1) \\ \vdots \\ b(m,n) \\ \cdots \end{pmatrix}$$

then probability covering can be defined as the following integer program

$$\begin{array}{rl} \min & \mathbf{T}^T.\mathbf{x} \\ s.t. & \mathbf{L} \cdot \mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \in \{0,1\}^D \end{array} \tag{4.7}$$

which can still be solved by branch-and-bound approach. This formulation can still integrate localized clutter, terrain masks and multi-mission constraints but cannot be combined with scan update rates. Probabilistic scan update rates would require to compute the probability of having at least $s(m,n)$ dwell detections in cell (m,n) which is written as a sum of products. Unlike single detection probability which is a single product, sum of products cannot be linearised using logarithm or anti-logarithm.

Conclusion and futures leads

Results and fallouts of the thesis

The paradigm shift of the digital era favoured the production of highly flexible radars, thanks to electronic scanning and digital processing. Dynamical beam-forming and beam-steering increase the degrees of freedom in designing radar search patterns, which can quickly shift between different beam-shaped radiation patterns.

Exploitation of those novel possibilities and efficient resource allocations are necessary as modern systems compete over shorter and shorter time frame in the context of electronic warfare. So far, little work has been done previously on the optimization of radar search patterns. Previous approaches limited the beam-shape or steering directions of dwell candidates for the radar search pattern. In the industry, the state of art are hand-designed patterns, requiring working time from engineers, and lacking situational adaptability.

The main challenge of this thesis was the identification of an appropriate theory for modelling radar scanning problems. This reflection has lead to the choice of combinatorial cover problems as a fitting basis for mathematical modelling. The reformulation of radar scanning from the perspective of combinatorial optimization provided a powerful theoretical framework for optimizing radar search patterns. It also proved to be a flexible tool, which has been extended to model complex situations with multiple mission requirements under localized constraints.

The thesis theoretical contributions to combinatorial optimization are the classification of radar cover problems in respect to complexity theory as either strongly polynomial-solvable or NP-hard problems, and the development and identification of optimization algorithms for solving those problems. More practical contributions also include the design of reduction methods for improving computational efficiency in solving radar cover problems, and the research on tools for generation and representation of multiple optimal solutions.

Under those considerations, the developed framework for optimization of radar search pattern has proven to be a powerful formulation, offering various opportunities. Beyond the academic possibilities, the present work also has potential industrial applications in computer-aided design of radar search pattern, where it can be used to generate first solutions for an existing radar which engineers could refine using their expertise. The automatic nature of the optimization algorithms presented in this thesis is also well-suited for simulation of future radar systems. The radar search pattern of different radar architectures could be optimized in parallel to compare their respective performances.

Short term applications focus on aided-design, but in the longer term, radar search optimization could be perform directly in operation, adapting the radar scanning mode to the situation parameters. Branch-and-bound is a practical method for generating just-in-time solution, which can be stopped at any time to return the best current solution. Knowing a lower bound on the optimal solution, thus having an estimation of the potential gain of pursuing optimization, is a useful feature for efficient radar resource management.

Futures objectives

The various advances made during this thesis have also brought questions and open the path for future research leads. The computational cost of the problem could be improve by modifying the grid quantification values, and thus the overall shape of the grid. A basic approach would be grid adaptation to the mission energetic requirements. More generally, this problematic falls into finite element analysis, a research field focused on discretization of smooth manifolds (“continuous spaces”) and their representation as finite meshes of elements. The discrete detection grid could in fact take any form, and does not require to be regular, or even rectangular. This is another strength of the proposed the framework: it separates the radar model from the combinatorial cover problem. The branch-and-bound method is very generic, and can be used regardless of the grid geometry. Informally, the algorithm only receives a discrete space, and a set of covers over this space to select, but is impervious to what the space actually represents.

This gain in computational efficiency could be used to extend the discretized space to higher dimensions. The detection grid presented in this work has only two dimensions, azimuth and elevation. However, radar detection is often considered in four dimensions: azimuth, elevation, range and Doppler. A four-dimensional grid would thus be able to account clutter not only from an energetic point view, but from a signal processing perspective, as it would discretize the spatial location of clutter, but also the speed range

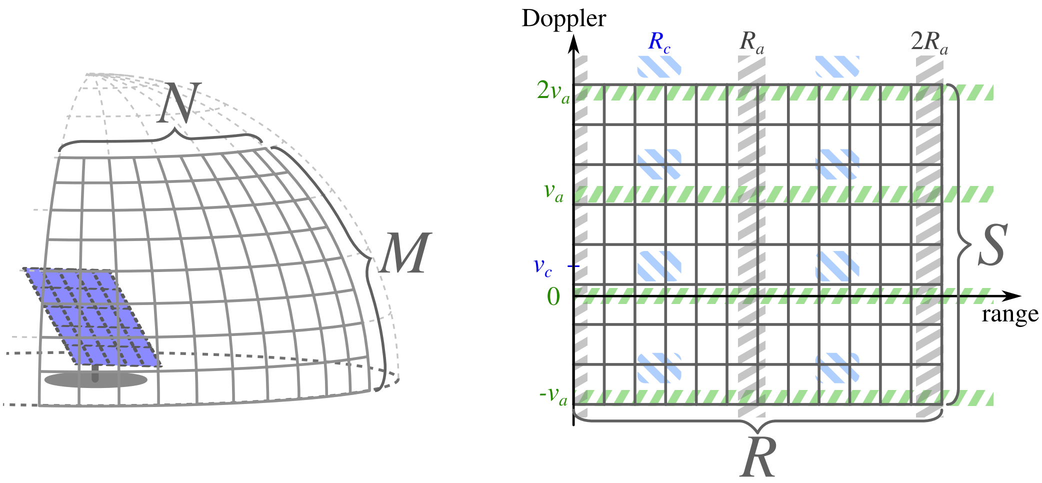

Figure 4.27: Spatial axis (left) and range-Doppler axis (right) of a four dimension M-by-N-by-R-by-S detection grid

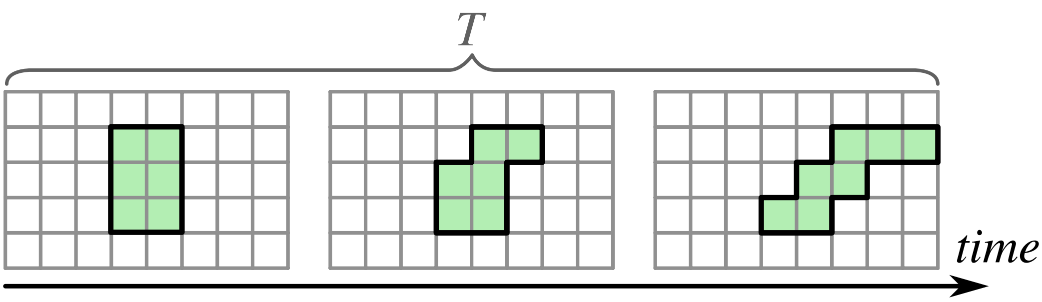

Figure 4.28: Time axis of a detection grid and dwell cover deformation due to radar movements

it pollutes, see Figure 4.27. Waveforms could be optimized as well in a four-dimensional detection grid model, by maximizing the waveform visibility, i.e. minimum number of visible dwells, respectfully to its burst parameters: period repetition interval, duty cycle, number of pulse, etc.

A fifth dimension could be added to account for time, see Figure 4.28, for example in settings with a frigate radar moving due to the ship yaw, pitch and roll. If the ship movement is regular enough and can be predicted, the radar search pattern could be optimized to compensate the radar movements. This would require to incorporate scheduling into the radar search pattern optimization.

While if the ship movement is irregular and noisy, it can be represented as a probability distribution. Probabilistic covering has been presented here for exploiting dwell overlaps. It could also serve to optimize the radar search

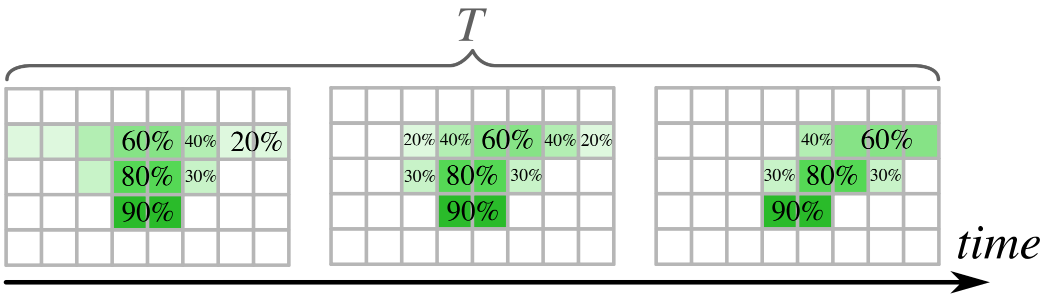

Figure 4.29: Dwell detection probability on the detection grid along the time axis under random radar movement

pattern in case where the radiation pattern is not fixed, but is displaced by a random shift due to the radar small erratic movements.

So there two main approaches to search pattern optimization of dynamically moving radars: a deterministic model of the radar movement or a statistical model. In fact both model could be combined: the radar movement could have a deterministic component, its average movements, to which a random part is added. This model could be solved on a fifth-dimensional grid with probabilistic covering, see Figure 4.29, where the detection probability on a cell would combine the waveform detection probability with the dwell presence probability under the radar random movement. Currently, those promising ideas are still being studied as directions for the future work succeeding this thesis.

In that aspect, the main, and most important result from this thesis is that combinatorial covering is a rich, powerful and flexible tool for modelling and optimizing radar search patterns.

Bibliography

[1] M. Skolnik, *Radar Handbook, Third Edition*. McGraw-Hill Education, 2008.

[2] W. Stutzman and G. Thiele, *Antenna Theory and Design*. Wiley, 2012.

[3] S. J. Orfanidis, *Electromagnetic Waves and Antennas*. Rutgers University, 2016. [Online]. Available: http://www.ece.rutgers.edu/~orfanidi/ewa/

[4] W. H. Von Aulock, "Properties of Phased Arrays," *Proceedings of the IRE*, vol. 48, no. 10, pp. 1715–1727, 1960.

[5] R. Brent, *Algorithms for Minimization Without Derivatives*, ser. Dover Books on Mathematics. Dover Publications, 1973. [Online]. Available: https://books.google.fr/books?id=6Ay2biHG-GEC

[6] B. R. Mahafza, *Radar Signal Analysis and Processing Using MATLAB*, 1st ed. Chapman & Hall/CRC, 2008.

[7] H. Meikle, *Modern Radar Systems*, ser. Artech House radar library. Artech House, 2001.

[8] P. Swerling, "Probability of detection for fluctuating targets," *IRE Transactions on Information Theory*, vol. 6, no. 2, pp. 269–308, 1960.

[9] D. P. Meyer and H. A. Mayer, "Radar target detection, handbook of theory and practice," *New York, Academic Press, Inc., 1973. 508 p*, 1973.

[10] V. V. Vazirani, *Approximation Algorithms*. Springer-Verlag New York, Inc., 2001.

[11] R. M. Karp, *Reducibility among Combinatorial Problems*. Boston, MA: Springer US, 1972, pp. 85–103. [Online]. Available: https://doi.org/10.1007/978-1-4684-2001-2_9

[12] D. S. Johnson, "Approximation algorithms for combinatorial problems," in *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '73. New York, NY, USA: ACM, 1973, pp. 38–49. [Online]. Available: http://doi.acm.org/10.1145/800125.804034

[13] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979. [Online]. Available: http://www.jstor.org/stable/3689577

[14] C. Lund and M. Yannakakis, "On the hardness of approximating minimization problems," *J. ACM*, vol. 41, no. 5, pp. 960–981, Sep. 1994. [Online]. Available: http://doi.acm.org/10.1145/185675.306789

[15] R. Raz and S. Safra, "A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np," in *IN PROC. 29TH ACM SYMP. ON THEORY OF COMPUTING, 475-484. EL PASO*, 1997.

[16] B. Escoffier and V. T. Paschos, "Completeness in approximation classes beyond apx," *Theoretical Computer Science*, vol. 359, no. 1, pp. 369 – 377, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397506003148

[17] A. Srinivasan, "Improved approximations of packing and covering problems," in *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '95. New York, NY, USA: ACM, 1995, pp. 268–276. [Online]. Available: http://doi.acm.org/10.1145/225058.225138

[18] T. Koch *et al.*, "Miplib 2010," *Mathematical Programming Computation*, vol. 3, no. 2, p. 103, Jun 2011. [Online]. Available: https://doi.org/10.1007/s12532-011-0025-9

[19] J. Beasley, "An algorithm for set covering problem," *European Journal of Operational Research*, vol. 31, no. 1, pp. 85–93, 1987.

[20] B. Yelbay, Ş. İ. Birbil, and K. Bülbül, "The set covering problem revisited: An empirical study of the value of dual information," *Journal of Industrial and Management Optimization*, vol. 11, no. 2, pp. 575–594, 2015.

[21] J. Beasley and P. Chu, "A genetic algorithm for the set covering problem," pp. 392–404, 1996.

[22] D. Krupa R., A. Basu Roy, M. De, and S. Govindarajan, *Demand Hitting and Covering of Intervals*. Cham: Springer International Publishing, 2017, pp. 267–280. [Online]. Available: https://doi.org/10.1007/978-3-319-53007-9_24

[23] J. Li and Y. Jin, *A PTAS for the Weighted Unit Disk Cover Problem*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 898–909. [Online]. Available: https://doi.org/10.1007/978-3-662-47672-7_73

[24] T. M. Chan and E. Grant, "Exact algorithms and apx-hardness results for geometric packing and covering problems," *Computational Geometry*, vol. 47, no. 2, pp. 112 – 124, 2014, special Issue: 23rd Canadian Conference on Computational Geometry (CCCG11). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925772112000740

[25] A. Schöbel, *Optimization in Public Transportation: Stop Location, Delay Management and Tariff Zone Design in a Public Transportation Network*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[26] G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, 1st ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999.

[27] J. Matouek and B. Gärtner, *Understanding and Using Linear Programming (Universitext)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[28] N. Megiddo, "On finding primal- and dual-optimal bases," *ORSA Journal on Computing*, vol. 3, no. 1, pp. 63–65, 1991. [Online]. Available: https://doi.org/10.1287/ijoc.3.1.63

[29] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York, NY, USA: Wiley-Interscience, 1988.

[30] Y. Briheche, F. Barbaresco, F. Bennis, D. Chablat, and F. Gosselin, "Non-uniform constrained optimization of radar search patterns in direction cosines space using integer programming," in *2016 17th International Radar Symposium (IRS)*, 2016.

[31] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*, 1st ed. New York, NY, USA: Cambridge University Press, 2011.

[32] M. Conforti, G. Cornuejols, and G. Zambelli, *Integer Programming.* Springer Publishing Company, Incorporated, 2014.

[33] F. Barbaresco, J. Deltour, G. Desodt, B. Durand, T. Guenais, and C. Labreuche, “Intelligent m3r radar time resources management: Advanced cognition, agility & autonomy capabilities,” in *Radar Conference - Surveillance for a Safer World, RADAR. International*, 2009.

[34] M. I. Jimenez, L. D. Val, J. J. Villacorta, A. Izquierdo, and M. R. Mateos, “Design of task scheduling process for a multifunction radar,” *IET Radar, Sonar Navigation*, vol. 6, no. 5, pp. 341–347, June 2012.

[35] S. L. C. Miranda, C. J. Baker, K. Woodbridge, and H. D. Griffiths, “Comparison of scheduling algorithms for multifunction radar,” *IET Radar, Sonar Navigation*, vol. 1, no. 6, pp. 414–424, Dec 2007.

[36] P. W. Moo, “Scheduling for multifunction radar via two-slope benefit functions,” *IET Radar, Sonar Navigation*, vol. 5, no. 8, pp. 884–894, Oct 2011.

[37] P. M. Hahn and S. D. Gross, “Beam shape loss and surveillance optimization for pencil beam arrays,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-5, no. 4, pp. 674–675, 1969.

[38] D.-S. Jang, H.-L. Choi, and J.-E. Roh, “Search optimization for minimum load under detection performance constraints in multi-function phased array radars,” *Aerospace Science and Technology*, vol. 40, pp. 86–95, 2015.

[39] S. Torres, R. Adams, C. Curtis, E. Forren, D. Forsyth, I. Ivic, D. Priegnitz, J. Thompson, and D. Warde, “A demonstration of adaptive weather surveillance and multifunction capabilities on the National Weather Radar Testbed Phased Array Radar,” in *Radar Conference (Radar), 2014 International*, 2014.

[40] P. Berman, J. Jeong, S. P. Kasiviswanathan, and B. Urgaonkar, “Packing to angles and sectors,” in *Proceedings of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures*, 2007, pp. 171–180.

[41] F. Y. Chin, H.-F. Ting, and Y. Zhang, “Variable-size rectangle covering,” in *Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications.* Berlin, Heidelberg: Springer-Verlag, 2009, pp. 145–154.

[42] B. D. Carlson and D. Willner, "Antenna pattern synthesis using weighted least squares," in *IEE Proceedings H (Microwaves, Antennas and Propagation)*, vol. 139. IET, 1992, pp. 11–16.

[43] G. K. Mahanti, A. Chakraborty, and S. Das, "Phase-only and amplitude-phase only synthesis of dual-beam pattern linear antenna arrays using floating-point genetic algorithms," *Progress In Electromagnetics Research*, vol. 68, pp. 247–259, 2007.

[44] Y. Han and C. Wan, "Scalable Alternating Projection and Proximal Splitting for Array Pattern Synthesis," *International Journal of Antennas and Propagation*, 2015.

[45] "IBM ILOG CPLEX Optimization Studio, v12.6," http://www-03.ibm.com/software/products/en/ibmilogcplestud/, 2015.

[46] S. Umetani and M. Yagiura, "Relaxation heuristics for the set covering problem," *Journal of the Operations Research Society of Japan*, vol. 50, no. 4, pp. 350–375, 2007.

[47] A. Caprara and M. Fischetti, "A Heuristic Method for the Set Covering Problem," *Operations Research*, vol. 47, no. 5, pp. 730–743, 1999.

[48] A. Caprara, P. Toth, and M. Fischetti, "Algorithms for the set covering problem," *Annals of Operations Research*, vol. 98, no. 1, pp. 353–371, Dec 2000. [Online]. Available: https://doi.org/10.1023/A:1019225027893

[49] E. Danna, M. Fenelon, Z. Gu, and R. Wunderling, *Generating Multiple Solutions for Mixed Integer Programming Problems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 280–294. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-72792-7_22

[50] E. Danna and D. L. Woodruff, "How to select a small set of diverse solutions to mixed integer programming problems," *Oper. Res. Lett.*, vol. 37, no. 4, pp. 255–260, Jul. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.orl.2009.03.004

[51] S. Mecke and D. Wagner, *Solving Geometric Covering Problems by Data Reduction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 760–771. [Online]. Available: https://doi.org/10.1007/978-3-540-30140-0_67

[52] P. Greistorfer, A. Løkketangen, S. Voß, and D. L. Woodruff, "Experiments concerning sequential versus simultaneous maximization of objective function and distance," *Journal of Heuristics*, vol. 14, no. 6, pp. 613–625, Dec 2008. [Online]. Available: http://dx.doi.org/10.1007/s10732-007-9053-z

[53] J.-F. Tsai, M.-H. Lin, and Y.-C. Hu, "Finding multiple solutions to general integer linear programs," *European Journal of Operational Research*, vol. 184, no. 2, pp. 802 – 809, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221706011611

UNIVERSITE BRETAGNE LOIRE

CENTRALE NANTES

Thèse de Doctorat

Yann BRIHECHE

Optimisation du maillage de la veille sur radar à balayage électronique à panneau fixe

Optimization of search patterns for fixed-panel tridimensional scanning radars

Résumé

Les radars modernes sont des systèmes complexes. Leurs missions, incluant surveillance, suivi et identification, se sont étendues conjointement à leurs capacités, favorisées par le développement de l'électronique et du numérique. Ces radars peuvent balayer dynamiquement et librement l'espace grâce à des panneaux numériques, les libérant des limitations des moteurs mécaniques. La guerre électronique, où les temps de réaction sont toujours plus courts, nécessite néanmoins une gestion parcimonieuse du temps disponible au radar pour accomplir ces missions.

Dans ce contexte, l'optimisation du temps utilisé pour la surveillance doit exploiter pleinement les capacités des nouveaux radars. Les travaux réalisés durant cette thèse ont été de formaliser mathématiquement ce problème, de déterminer et adapter les outils pertinents pour sa résolution, et d'en explorer les possibilités. Le problème de la surveillance radar se rapproche conceptuellement du recouvrement d'ensemble en optimisation combinatoire. Grâce à des algorithmes utilisant la programmation dynamique et la programmation linéaire en nombres entiers, ce problème a pu être résolu, et étendu à des situations plus complexes, incluant différentes contraintes opérationnelles.

Cette approche fructueuse ouvre de nouvelles pistes pour l'amélioration des performances des radars, et offre de nombreuses possibilités d'applications. Entre autres l'aide à la conception des couvertures des radars actuels, la simulation des performances d'architectures de futurs radars et le développement de radars cognitifs, capables de s'adapter à leur environnement opérationnel.

Mots clés
gestion des ressources radar,
radar à balayage électronique,
antenne réseau à commande de phase,
optimisation combinatoire,
recouvrement d'ensemble,
problème de recouvrement de grille rectangulaire

Abstract

Modern radars are complex systems, capable of multiple functions: scanning, tracking, identification, etc. With the advent of electronic and digital technologies, radars can dynamically and freely sweep their surroundings using fixed-panels, freeing them from the limitations of mechanical rotation. With increasingly intelligent and adaptable systems competing in modern warfare in ever shorter time, careful management of the radar available time-budget is required to achieve desired performances and ensure civilian and military safety.

In this context, optimization of radar search pattern time-budget must exploit modern radars full potential. This thesis main accomplishments are the mathematical modelling of radar search pattern optimization, the identification and development of appropriate tools for its solving, and the exploration of the model possibilities. Radar search pattern design can be related to covering problems in combinatorial optimization. Radar covering can be solved using methods based on dynamic programming and integer programming, and can furthermore be extended to account for more complex situations with multiple operational constraints.

The tools developed in this thesis provide a powerful and flexible framework for solving radar covers problems. This framework opens interesting research avenues for improving radar performances. It offers various possible applications for aided-design of radar search patterns, simulation of new radar architectures performances, and development of cognitive radar systems capable of adapting in real time to the operational environment.

Key Words
radar resource management,
tridimensional radar,
phased array antenna,
combinatorial optimization,
set covering,
rectangular grid cover problem

UNIVERSITÉ BRETAGNE LOIRE