On the optimal placement of cameras for the surveillance of urban events : a real-world, human-assisted combinatorial approach for decision support systems

Julien Kritter

IRIMAS UNIVERSITÉ HAUTE-ALSACE
Institut de Recherche en Informatique, Mathématiques, Automatique et Signal

On the optimal placement of cameras for the surveillance of urban events:

A real-world, human-assisted combinatorial approach for decision support systems

A report submitted in partial fulfilment of the degree of Doctor of Philosophy in the field of Computer Science by

Julien Kritter

PhD thesis report

Director	Pr. Lhassane Idoumghar
Advisors	Dr. Mathieu Brévilliers
	Dr. Julien Lepagnot
Reviewers	Pr. Edward Keedwell
	Pr. Abderrafiaa Koukam
Viva committee	Pr. Sébastien Verel
	Pr. Clarisse Dhaenens
	Pr. Pierre Collet

UNIVERSITÉ HAUTE-ALSACE

Thanks

While it is often said—rightfully so perhaps—that writing a PhD thesis is a naturally solitary enterprise to undertake, I can think of quite a handful of people whose support it would be incredibly naive of me to belittle. Before we begin with the subject matter of this report, I would therefore like to take a moment for gratitude, and sincerely apologise in advance for any mention I may forget to make.

I will begin, as I believe is customary, with the people who have been closest to me professionally during the last three years. I would first like to thank Dr. Mathieu Brévilliers, for his continuous availability and unwavering rigour, both of which I have found to be very valuable throughout this project. Now, of course, there were four of us around this thesis, and I must therefore also mention Dr. Julien Lepagnot, the coordinator for the OPMoPS project, for his guidance, as well as Pr. Lhassane Idoumghar, the supervisor for my thesis and, in fact, our entire research team. I thank you both for your support and for making this thesis possible.

As the people close to me will no doubt attest, it would have been atypical of me to undertake such a project for any reason other than passion and sheer intellectual entertainment. While I believe I have lived with such spirit for a while now, there are several people I would like to thank for having been able to cultivate my eagerness to learn. I will begin with retired Pr. Bruno Taconet from the University of Le Havre–Normandy. Your commitment to passionate teaching and to your values as an academic has been a steady and, at times, much needed source of hope. From the same institution, I must also thank Dr. Cédric Joncour, Dr. Sophie Michel-Loyal and Dr. Xavier Schepler for taking me under their wing during my Masters thesis. The field you have introduced me to truly fascinates me and I cannot thank you enough for having done so with such a refreshing approach to academic research. I also want to mention Dr. Laurent Amanton, Pr. Eric Sanlaville, Dr. Antoine Dutot, Dr. Claude Duvallet, Pr. Damien Olivier, Pr. Frédéric Guinand, Dr. Stefan Balev, Dr. Yoann Pigné, Dr. Véronique Jay and Dr. Abderrazak Zahour. Learning from you all has more than once put a smile on my face, and I am grateful to have been taught by so many truly passionate individuals.

Still close to the workplace, I must also thank the colleagues and fellow PhD students I have worked with these last three years. The latter have included Dr. Hojjat Rakhshani, soon-to-be Drs. Soheila Ghambari, Mounir Bendali-Braham, Cassandra Rotily and Imène Zaidi, Dr. Mokhtar Essaid as well as M. Maxime Pinard. Among my colleagues, I also thank Dr. Laurent Moalic and Dr. Dominique Schmitt for their support on a couple of side research projects. I must also mention Pr. Mahmoud Melkemi and, of course, Dr. Karine Zampieri, who has been a great guide to me as we were giving basic algorithmics classes together, a first for me.

To conclude on a more personal level, there are a couple of people I simply cannot forget about. I will quite naturally begin with my mother for her unconditional love and support, and without whom this thesis would have ended prematurely. Finally, to friends, all of whom I shall mention through nicknames, for only they need to recognise themselves. Thank you, then, to my fellow gunpowder plotter, to her Ladyship, and to all of you from Uni: the enchanter, the other Julien, the other guy with a hat, the dessert, the modern hacker and all the others! You have all helped make this journey quite fun, and I hope I can keep up with you for yet another couple of years.

And now... to surveillance!

Summary

As part of a French-German joint research project, this thesis tackles issues related to video-based surveillance networks, more specifically their design in terms of optimal camera placement and the integration of user input for deployment in the context of decision support systems. We begin with an in-depth review of two bodies of literature, one related to our practical application and the other to a popular and closely related theoretical problem. This survey covers both problem modelling and solving, and highlights several open lines of research. In terms of modelling, we begin by focusing on the design of a middle ground between unrealistic simplicity and prohibitive complexity, both popular extremes in optimal camera placement literature. We design an instance generation and preprocessing framework which uses real-world data to extract accurate information about our surveillance area. Taking advantage of the availability of such instances, we then move on to benchmarking the state of the art, a study which had so far not been performed, most likely due to the large number of problem variants and constraints in the literature. Our results suggest several hypotheses on the complexity of the problem, which we investigate further. Our conclusions lead us to the construction of a new model for our problem, which we then use to design a time-efficient solving algorithm. The latter is then integrated into a small decision support system through which a user can negotiate solutions with the solver until the output is deemed satisfactory.

Résumé

Cette thèse, contribution à un projet de recherche franco-allemand, s'intéresse aux réseaux de vidéo-protection, plus particulièrement au problème de placement optimal de cameras et à l'intégration de cette problématique au sein de systèmes d'aide à la décision. Nous commençons avec une revue approfondie de deux corpus, l'un portant sur notre application et l'autre sur un populaire problème théorique lié à cette dernière. Cette étude couvre à la fois modélisation et résolution, et met en avant diverses pistes de recherche. En terme de modélisation, nous nous intéressons particulièrement à la mise au point d'un compromis entre les deux extrêmes typiques de la littérature : trop simpliste ou trop coûteux. Nous implémentons ainsi un système de génération et de pré-traitement d'instances basé sur des données réelles et capable d'extraire une représentation fidèle de la zone à couvrir. Ces instances nous permettent ensuite de mettre en place une étude comparative de l'état de l'art, jusqu'ici resté sans point de référence en raison du large panorama de variantes et de contraintes autour de notre problème. Nos résultats posent plusieurs hypothèses au sujet de la complexité de ce dernier. Nos conclusions associées nous permettent de construire un nouveau modèle, que nous utilisons afin de définir un algorithme de résolution efficace, et directement intégrable à un système d'aide à la décision. Un prototype est d'ailleurs implémenté afin de permettre à un utilisateur de négocier avec l'algorithme jusqu'à obtenir une solution jugée satisfaisante.

Contents

Introduction

The OPMoPS project

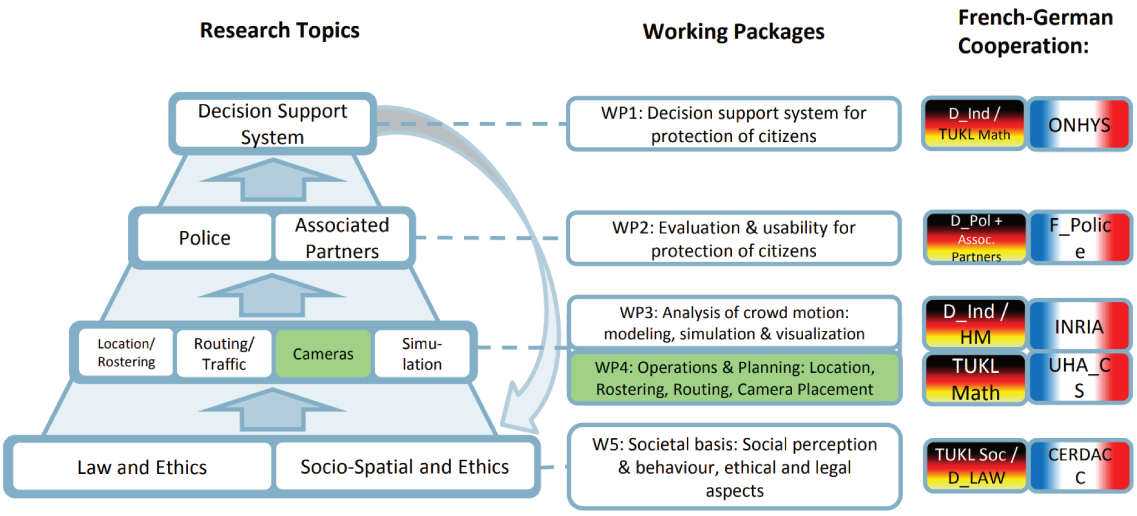
This thesis is part of a joint research project called OPMoPS which brings together the French Agence Nationale de la Recherche (ANR) and the German Bundesministerium für Bildung und Forschung (BMBF) on the subject of organised pedestrian movement in public spaces. The project focuses on the preparation and crisis management for urban parades and demonstration marches with high conflict potential. It was launched in 2017 and is currently coordinated by Dr. Julien Lepagnot from the Université de Haute-Alsace, France. The official translated summary of the project is given below.

> Parades of highly controversial groups and political demonstration marches are considered a major threat to urban security, since the diametrically opposed opinions of the participants and opponents may result in violence or even terror attacks. Due to the movement of the urban parades and demonstration marches (in the following abbreviated by UPM) through large parts of cities and the resulting space and time dynamics, it is particularly difficult for forces of civil security (abbreviated in the following by FCS) to guarantee safety at these types of urban events without endangering one of the most important indicators of a free society. In this proposal, partners representing the FCS (police and industry) will cooperate with researchers from academic institutions to develop a decision support tool which can help them both in the preparation phase and crisis management situations of UPMs. The development of this tool will be driven by the needs of the FCS but also by research results on the social behaviour of the participants and opponents. The latter and the assessment of legal and ethical issues related to proposed technical

> solutions are an important part of the proposed research. Specific technical issues which the French-German consortium will have to include the following: optimisation methods to plan UPM routes, transportation to and from the UPM, location and personnel planning of FCS, control of UPMs using stationary and moving cameras, as well as simulation methods, including their visualisation, with specific emphasis on social behaviour. The methods will be applicable to the preparation for and organisation of UPMs as well as to crisis management for ad-hoc UPMs or unexpected events. At the end of the funding period a decision tool will be available which shows the potential of the approach and which will be marketable under a foreseeable amount of time and work.
>
> (ANR: Agence Nationale de la Recherche, anr.fr/Project-ANR-16-SEBM-0004)

The OPMoPS project brings together partners with various backgrounds in the fields of psychology, sociology, law enforcement, software engineering, mathematics, computer science, crowd dynamics simulation and law. The full list of institutions involved in this joint effort is given below, along with a diagram which illustrates the role of each partner and highlights ours.

- The University of Koblenz and Landau ;
- The Munich University of Applied Sciences ;
- The University of Kaiserslautern ;
- The University of Upper Alsace ;
- INRIA Rennes ;
- ONHYS SAS ;
- virtualcitySYSTEMS GmbH ;
- VdS Schadenverhutung GmbH ;
- The Rhineland-Palatinate Police School ;
- Le Centre de Recherche de l'École des Officiers de la Gendarmerie Nationale.



Thesis outline

One of the first questions which arise when designing video surveillance infrastructure is that of determining the appropriate positions and orientations for cameras such that our coverage and cost objectives are met. From a computational standpoint, this problem is widely recognised as complex, and a significant amount of research has been put into solving it efficiently. This interest has only been vivified in the last few years with the launch of a handful of defence projects by countries around the globe, all of them conscious of today's need for a better approach to global security.

In this thesis, we introduce a different approach to solving the optimal camera placement problem in such a context. While most of the literature has so far focused on defining clear operational constraints and aiming for optimality from there, it would appear that such approaches tend to be rather difficult to implement for town officials and law enforcement officers working on the field. Through discussions with our partners, including the aforementioned field experts, it has become clear that any new approach should be designed with a user at its centre. In other words, an ideal system would not provide just one solution, but rather would continuously take in feedback from a user and adjusting until a satisfying solution is found. We describe such an approach in this report.

We begin in Chapter I with a review of two bodies of literature: that of optimal camera placement, and that of the very closely related set cover problem. We then move on to Chapter II in which a new instance generation and preprocessing frame-

work is introduced to reach a compromise between the extremes typically found in the literature. Several hypotheses about the complexity of our problem are then suggested for Chapter III. This part of our study includes a benchmark of the state of the art as well as the confirmation of several observations made earlier. These serve as the basis for the design of our human-assisted optimisation approach, which we describe in Chapter IV. More detailed introductions can be found at the beginning of each chapter.

Chapter I

Literature review

Introduction

We begin by reviewing the current state of the art for both the optimal camera placement problem and its underlying set covering problem. We start with the former and introduce its origins as well as its influence in the field of computer vision. We quickly move on to surveillance applications, namely target tracking and, of course, global and persistent surveillance. We then focus on this last aspect and review the literature on surveillance area modelling, sensor representation and visibility analysis approaches. With these foundations in place, we proceed to studying solving methods, both deterministic and stochastic. We then move on to the popular set covering problem, the structure of which is fundamentally identical to that of basic camera placement models. We highlight the strong relationship between the two problems, a connection which we found was seldom made in the literature. This allows us to continue on to solving methods which we extensively review and compare while regularly referring to optimal camera placement when possible.

Related publication

[119] Julien Kritter et al. "On the optimal placement of cameras for surveillance and the underlying set cover problem". In: *Applied Soft Computing* 74 (Jan. 2019), pp. 133–153. DOI: 10.1016/j.asoc.2018.10.025

I.1 Origins and introductory applications

The optimal camera placement problem finds its root in a famous geometry problem dating back 1987: the art gallery problem (AGP). Seminal work on the AGP is credited to O'Rourke for his contribution *Art Gallery Theorems and Algorithms* [153], although earlier references can be found, namely work by Klee (published by Honsberger [94]) and Chvátal [39]. An art gallery is given as a two-dimensional **simple polygon** which corresponds to the building's floor plan. The goal is to place **guards** inside the gallery such that for any point inside the polygon, there is at least one guard to which it can be connected by a segment without intersecting with any of the polygon's edges. We say that the polygon's area must be entirely covered by the guards (Figure I.1). It has been shown that for any polygon with n vertices, the optimal solution (which minimises the number of guards) requires no more than $\lceil \frac{n}{3} \rceil$ guards, all of which can be placed on the polygon's vertices [74, 39]. Several variants of the problem exist such as the **watchmen route problem** [36], which uses mobile guards, or the **floodlight illumination problem** [22], in which light sources replace the guards. For more information about these variants, the reader is referred to [106, 31, 61, 65, 181, 86].

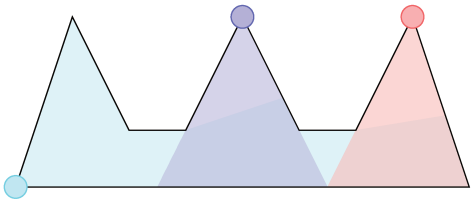


Figure I.1: An instance of the Art Gallery Problem with $n = 9$, and a solution with 3 guards

The optimal camera placement problem is of course a much more constrained variant of the AGP. To begin with, cameras do not have unlimited range. In fact, due to image quality requirements which are usually enforced by applications, the range of a camera is often much more limited than one might expect. Furthermore, no camera model can achieve 360-degree visibility, although some can achieve something very close, as we will see later.

Computer vision is one of the most active fields of research when it comes to optimal camera placement, and has introduced many of the constraints and objectives used in modern models. This is particularly true with **photogrammetry** problems for scene or object reconstruction from camera images. In this case, finding the **next**

best view [167] is particularly important, and an optimal placement of the cameras is essential to achieve high-quality solutions and remodels. A typical example of this problem can be found in work by Olague and Mohr [154]. For earlier and more generic work, the reader is referred to "A survey of sensor planning in computer vision" by Tarabanis, Allen, and Tsai [183].

Still in computer vision, **feature extraction** also appears to significantly benefit from optimal camera positions. In this case, strong image quality requirements are typically enforced, since the images produced by the cameras feed sensitive image analysis algorithms. Work by Fehr, Fiore, and Papanikolopoulos [69] can be cited here, as it formalises not only camera **fields of view**, but also the elements which are to be covered and around which the images must be of sufficiently high quality. Distinctions are made between the requirements for **gait classification**, **facial recognition** and other image processing problems, and a new quality function taking into account issues such as **foreshortening**, ground coverage or resolution is defined. For more details about how such image-related constraints typically integrate into optimisation models, the reader is referred to a 2016 survey by Liu, Sridharan, and Fookes [125].

Regarding surveillance applications, target tracking is a very popular problem, which often benefits from progress in both computer vision and optimisation. Work by Ercan et al. [62] tackles one of the problem's variants: **target localisation**. The authors therefore attempt to locate objects in a given environment while attempting to minimise communication costs. A recurrent idea in such application is that of **hand-off rates**. This new constraint forces optimisation algorithms to not only cover the targets but also to ensure there is always a coverage overlap between two neighbouring cameras (see Figure I.2). This ensures that when a target reaches the edge of one camera's **frustum**, it also enters that of another to secure the hand-off and ensure the network never loses track of the target.

Authors for such an approach include Bodor, Schrater, and Papanikolopoulos, who worked from a set of predefined target trajectories in a given surveillance perimeter, and optimised the network layout so as to maximise image quality for motion coverage along these paths [19]. This work has been extended in [20] so as to include mobile cameras, and in [148, 149], Natarajan et al. used partially observable Markov processes to control the orientation of cameras as targets move freely across the surveillance area. For this approach, Fusco and Gupta [78] had previously worked on target coverage, and defined the objective function as the minimisation of each point's dark (uncovered) time. A similar application has also been studied by Konda and Conci [116], who used the cameras' feeds to switch them from global area coverage to target tracking and back, depending on whether or not individuals were being

detected within the coverable perimeter.

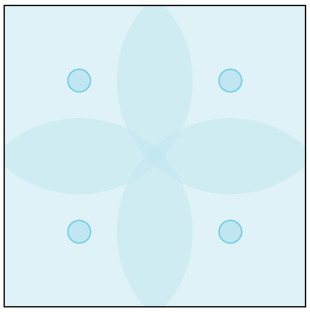


Figure I.2: Hand-off margins in a generic sensor network. Objects may leave the range of any sensor but always remain visible to the network

When it comes efficient tracking, static camera infrastructures can often be seen as insufficient. Targets can indeed be unpredictable and unless there are many of them, most of the infrastructure will likely be inactive most of the time. To better tackle this issue, modern literature has started to include mobile cameras. In real-world applications, these often represent **UAVs** (unmanned aerial vehicles), although some camera models can still **pan** and **tilt** after setup while having a fixed mounting point. For this, references include Bodor et al. [20] who positioned cameras so as to maximise activity detection accuracy in a given area, and Natarajan et al. [148, 149] who allowed the cameras to rotate when necessary.

Bringing UAVs into the model, an early search-theoretic approach to the problem can be found in [10], in which Baum and Passino split the search area into smaller regions and minimise the time required to cover the whole environment with a given fleet of UAVs. Allowing for some a priori probabilistic data about target distribution, Flint, Polycarpou, and Fernandez-Gaucherand [76] lay out a dynamic programming algorithm for path planning which maximises information gain for static targets. Considering both known and unknown target locations, Sinha, Kirubarajan, and Bar-Shalom proposed a fully decentralised algorithm with which UAVs are able to track known targets while detecting unknown ones, and share available information with other drones in range, for later path planning [173]. Others authors include Tang and Ozguner [182], who focus on information age, and minimise the overall time between two observations of a target in the environment. A gradient algorithm is used when only one UAV is available, and a decentralised implementation tackles the problem with a fleet. At this stage, a 2006 review of heuristics for coordinated target tracking was published by Wise and Rysdyk in [187]. Later work along those lines involves Kim and Kim [110], who consider single target tracking with multiple

UAVs in an occluded environment. Starting from a circular path around the target, the algorithm adjusts the route so as to take into account buildings for occlusion, and ensures that the target is visible by at least one drone at all times. This work was inspired by Rafi et al. [161] and Peot et al. [158], although the environment is free of possible occlusion in the former, and the latter is limited to static objects. An extension to evasive targets was later suggested by Kim and Crassidis [109]. In a simpler environment, Ding, Rahmani, and Egerstedt also published work related to target (convoy) tracking when the trajectories are known and paths may be planned in advance [55]. More recently, Zorbas et al. made additional contributions on this problem [201, 160, 202] and used linear programming as well as greedy heuristics to minimise both fleet size and energy consumption. Extensions have allowed for moving targets using time windows, and clustering methods to reduce problem size. In less deterministic conditions, work by Capitán, Merino, and Ollero involving partially observable Markov processes may also be mentioned here [28].

I.2 Camera placement for global and persistent surveillance

Overview

In the context of surveillance applications, global area coverage is certainly among the most studied problems in the literature. It is typically concerned with the placement of cameras across a given surveillance area following one of two strategies: (1) to maximise coverage with a limited number of cameras or (2) to minimise cost, or the number of cameras, under the constraint that no region of the surveillance area should remain unwatched. One of the first papers to consider complete area coverage while minimising network cost was authored by Horster and Lienhart [95], although Erdem and Sclaroff [63] provide an applicable and more general approach. These papers were later frequently referenced for further research and more specific applications. This is the case in work by David, Idasiak, and Kratz [53], which minimises the cost per covered unit, therefore relaxing the complete coverage constraint. Another variant can be found in one of two problems modelled by Murray et al. in [145], for which exactly p cameras must be distributed over regions of varying importance. A simpler model for this problem, without the network size constraint (p), was also proposed by Yabuta and Kitazawa [190]. More generally, the idea of partitioning the area into weighted regions (and defining so-called essential regions) is very recurrent, [41, 98] being two more examples.

The use of the hand-off rate (see Section I.1) in fitness evaluation has also appeared in area coverage problems. Murray et al. [145] derived a multiobjective backup coverage location problem from [52] in order to aim at a trade-off between primary coverage, with at least one camera, and backup (or overlapping) coverage, supported by at least two cameras at any time. The problem's reduction to a single objective formulation using the weighted sum method highlights the contradictory nature of the two goals. For further work on this case, the same authors proposed another solving approach in [111]. A study of the impact of hand-off rates was performed by Yao et al. [193, 192], who compare their work to that of Erdem and Sclaroff [63]. Results must however be put into perspective as the solving methods are very different in nature.

Surveillance applications with a clearer focus on security have also emerged lately, adding constraints on network robustness or defining the objective function in terms of how efficient the solution would be in case of an emergency. With this in mind, Morsly, Djouadi, and Aouf [140] define the best interceptor placement problem, for which cameras are set up to maximise the efficiency of human agents, should an intruder be detected. For a variant, Konda and Conci [115] consider the possibility of network reconfigurations in scenarios involving the sudden shutting down of cameras. With similar concerns, Rebai et al. [163] adjust the typical problem formulation to add so-called network protection constraints according to which a feasible solution must ensure that cameras watch each other as well as the surveillance area. With a different end goal, Zhang et al. have also approached the problem with connectivity constraints in [196], and suggest the design of a collaborative network of nodes for information processing. Beyond the specifics of camera networks, such issues are often encountered when positioning wireless sensors: recent work along those lines includes [162, 164]. Finally, although it does not address security itself, work from Zhao and Cheung [198, 199] may also be mentioned here, as it addresses privacy concerns. Using methods previously studied by the authors [35], the approach optimises the placement of cameras in order to ensure the cutting-out and inpainting of preselected individuals wearing visual tags on their chests.

Extending on the mention of mobiles cameras in Section I.1, a smaller amount of research has also been conducted on area surveillance with UAVs. Early work has involved Girard, Howell, and Hedrick [82], with an application to border or perimeter control. A five-layer implementation spans across operation planning and low-level control. Considering dynamic perimeters such as those emerging in forest fires and oil spills, Kingston, Beard, and Holt also tackled the problem and allowed for perimeter variation and fleet reinforcement [113]. This work was later specialised for road surveillance in [112]. More generically, another approach was later suggested

by Savla, Bullo, and Frazzoli [170]. Rather than to seek coverage of the entire area, the authors used the facility location problem as a model and minimised the worst-case travel time required by any UAV to reach any given point in the environment. For persistent surveillance, work by Nigam and Kroo can be mentioned instead, as the authors consider the problem of minimal information age across a sampled surveillance area [151, 152]. Using 3D models as a basis for space representation, Cheng, Keller, and Kumar extended typical mission planning work and proposed an algorithm which ensures that the trajectory followed by a UAV covers all buildings in the area at one time or another [34]. Regarding dynamic camera representation, work by Ahmadzadeh et al. may also be referenced here, as it tackles a coverage problem through path planning, and considers the changes in the field of view of a UAV as it banks and turns [2]. A distributed approach to coverage was later studied by Schwager, Julian, and Rus [171], although the paper only considers simple two-dimensional polygons as a potential surveillance area.

Surveillance areas

The following sections will provide more details about each of the core aspects of optimal camera placement. They will focus on our application, that is, area coverage or global surveillance. We begin with the modelling of the surveillance area.

In the art gallery problem (see Section I.1), the surveillance area is modelled using a two-dimensional polygon. The coverage area is therefore continuous and the solution's feasibility is measured by intersecting the guards' ranges with the gallery. While this may appear to be the most straightforward approach, there has been, to the best of our knowledge, no significant study of optimal camera placement in continuous space. In every reference mentioned in this review, surveillance areas are modelled using discrete components. This may be explained by several reasons. The first, computational, is that a realistic environment model would likely require the use of complex geometrical constructs, for which computations such as area and volume intersection are either unknown or prohibitively expensive. Another motivation comes from the very nature of the optimisation methods used, which often require the definition of a discrete solution neighbourhood. In any case, while often being computationally more expensive, the discrete approach also displays some important realistic elements for both space and cameras, such as the fact that the latter simply cannot be placed anywhere in space to begin with.

In order to bring the problem into the discrete domain however, a **sampling procedure** must be designed. The literature includes several options, most of which work in two-dimensional space, as is the case in the AGP. Early work can be at-

tributed to Erdem and Sclaroff [63], and Horster and Lienhart [95], who used linear integer programs which cannot work without discrete points to cover. The approach here is quite simple: a sampling frequency is chosen and applied to both the x- and y-axes to yield a discrete grid of points. An example of this approach can be found in Figure I.3. It should be noted that while a very significant amount of research has used this model [5, 53, 198, 193, 199, 140, 37, 164], it has in fact been extended to three-dimensional spaces by Andersen and Tirthapura [4].

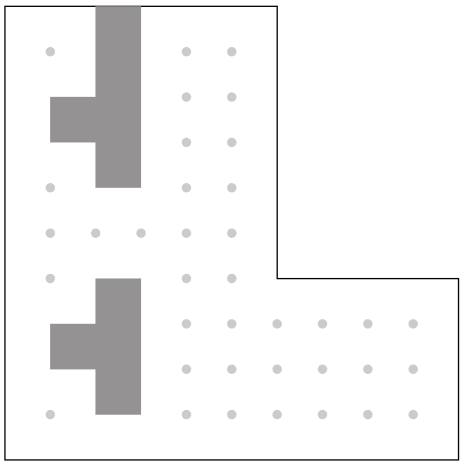


Figure I.3: A sampled floor plan, similar to the exhibition hall example found in several papers, with example coverage points

Other approaches for space sampling include the use of square rectangular grids, as is the case in [145, 111] : for their work on both primary and overlapping coverage, Murray et al. lay a regular grid on the surveillance area and consider each cell as a unit of coverage demand, to be satisfied by at least one camera. Similar work can be found in [41, 196, 77]. In their privacy-aware work [198], Zhao and Cheung reuse the work done by Horster and Lienhart but use dynamic sampling frequencies, adjusted based on the results of the optimisation process so far. Yabuta and Kitazawa [190] have chosen to sample their floor plan by extending wall lines to draw large, variable-sized, weighted rectangular regions, the centres of which act as the points to cover. Although this method appears to simplify the sampling process, it does come at the cost of a lesser degree of accuracy and applicability to arbitrary environments. A similar model was used by Indu et al. [98], who sample the 3D surveillance area by extracting rectangular boxes of varying importance.

At the other end of the spectrum comes a set of methods which use the full strength of 3D modelling and rendering software to evaluate the quality of a solution.

In [101], Janoos et al. define the environment using a detailed triangular mesh, which includes a ground plane to surveil and several types of structures which may cause occlusion. The concept of region importance is also brought forward through the use of so-called saliency and activity maps, which respectively quantify the structural importance of a point, and the amount of human activity recorded on it.

Camera representation

Next comes the matter of modelling the coverage of a single camera. This effectively provides us with the required geometrical data to determine which of the **ground samples** are covered by placing a camera in a given configuration. The actual process will be introduced in the next section.

Although space can be modelled in both 2D and 3D using rather similar approaches, the representation of a camera's viewing frustum (the polytope in which all points are visible to the sensor) has taken many forms in the literature, with good reasons. To begin with, it is worth considering three types of cameras: static, pan-tilt-zoom (PTZ) and omnidirectional. Static cameras are placed and orientated once using the values output by the optimisation method. PTZ cameras are initialised in the same way, but can later be instructed to pan (vertical axis), tilt (horizontal axis), zoom, or any combination of those within given ranges. Omnidirectional cameras are static, but have a distorted torus-shaped viewing frustum, with 360-degree coverage around their mounting points. In two dimensions, these cameras are usually defined by their sole position (x, y), while static and PTZ cameras, which need to be oriented, take the form of a 5-tuple $(x, y, \phi, \theta, \zeta)$ which includes the pan, tilt and zoom parameters, in that order. In 3D, a z parameter is simply added. Throughout this document, we will use the term **camera candidate** to represent those 5-tuples.

In two dimensions, Horster and Lienhart [95] have chosen to use simple isosceles triangles, with the vertex angle on the camera, to represent a sensor's coverage of the ground plane. Visibility analysis can then be conducted in advance by running a **point-in-triangle test** for every ground sample, and every possible assignment of the aforementioned camera parameters. To allow for such an approach, parameters ϕ, θ and ζ must be subjected to the same sampling procedure as the environment. As was the case for these authors' space models, this representation has been reused in several papers including [193, 140, 192] and [87] which also includes omnidirectional cameras. For a variant, Yabuta and Kitazawa [190] slightly adjust the representation by replacing triangles with circular sectors, which tend to be more recurrent in early 2D-oriented literature [148, 143].

While extending the problem from two to three dimensions doesn't strongly affect

space representation, it does raise a problem for camera modelling. A paper by Zhang et al. [196] actually highlights this problem: if a camera, placed at a given height, is modelled as a triangle or circular sector, then a blind spot is created right under the camera's nose (Figure I.4). A small area within the sector, near the camera, is in fact not visible in practice. Given the camera parameters given above, and technical specifications (resolution, focal length or Charge-Coupled Device width and height [77]), it is possible to use projection matrices to compute the coordinates of a parallelepipedic region ahead of the camera's nose, which delimits the true section of the ground plane visible by a camera. Work using this approach has included [69], in which Fehr, Fiore, and Papanikolopoulos use the projected region, taking into account the camera's height, to treat the problem in two dimensions. A similar projection is used in [140], although in this case the cameras are set up looking straight down from the ceiling, which eliminates any possible blind spot as far as ground coverage is concerned. Konda and Conci have also used this approach for their work towards network robustness [115] and later for a tracking problem [116]. Zhang et al. also suggested the use of a simpler model using a cone in [196] (reused in [164]), which therefore projects a disk rather than a polygon on the ground plane. While it does solve the blind spot problem, this representation is less accurate as far as the camera's field of view is concerned. A 2012 survey by Mavrinac and Chen which surveys geometric models for coverage can be found in [134].

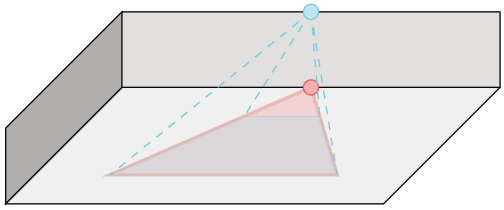


Figure I.4: The blind spot problem created by 2D space representation and 3D camera placement. A part of the red area is covered in the 2D model, but becomes invisible as the camera is placed in practice

Visibility analysis procedures

As was mentioned in the previous section, defining the camera model's representation allows us to determine which ground samples are seen or covered by which camera candidates. The methods used to actually perform this computation for a given

camera model are numerous, and optimal camera placement literature includes a good number of them.

In 2D, simple point-in-triangle or point-in-sector algorithms may be used, and extended to 3D (point in cone, point in pyramid). While these methods show several advantages, including straightforward parallelisability, it is worth mentioning some authors who have decided to use more applied approaches. They include Angella, Reithler, and Gallesio [5], who used OpenGL [88] depth maps, and a potentially expensive ray tracing algorithm, to determine whether an object or point is visible. Hengel et al. use a similar approach in [91].Fantini and Chaimowicz [68], on the other hand, delegated the modelling work and visibility analysis to full-blown game engines, namely Half-Life and Irrlicht, although these are closer to the original AGP formulation in that they model human vision (in first-person shooter games) rather than camera coverage. Another technique may also be found in work by Murray et al. [145], who used GIS technology to compute each camera's viewshed (a more generic definition for the viewing frustum). In spite of their higher computational cost, an advantage of these approaches is that they are able to tackle problems which arise in realistic environments such as occlusion. Targeting that problem, in [138], Mittal and Davis detail the formulation of a probabilistic framework for dynamic occlusion caused by moving objects (humans), which was integrated into their tracking tool, M2Tracker [136]. An application of the approach can be found in [137] by the same authors, and an insight into how static occlusion may be tackled in complex environments has been given in [101] by Janoos et al.

Solving the problem deterministically

While most of the work in exact methods comes from a transformation to a known Karp problem [105] (see Section I.3), some deterministic approaches which address the camera placement problem are worth mentioning. Early work towards exact solving must be attributed to Erdem and Sclaroff [63] and Horster and Lienhart [95], who formulated the 2D camera placement problem using binary linear integer programs (BIPs). Horster and Lienhart formulated the problem in a straightforward fashion, including coverage constraints and minimising the size or cost (when several types of cameras were available) of the infrastructure. In the objective function (I.1) and its associated constraints (I.2), $x_{xy\varphi}$ is set to 1 when a camera is placed at (x, y) with orientation φ. The binary function c maps every camera candidate to the set of grid cells it is able to cover. The program was solved to optimality using LPSolve [17] on two space grids: 12x12 with 2 possibles orientations, and 8x6 with 16 possible orientations. These configurations correspond to 288 and 768 camera candidates

(variables) respectively, and typically incur rather large, fast-growing search spaces. The paper concludes by highlighting the need for a more scalable approach.

$$\textbf{min} \sum_{x=1}^{s_x} \sum_{y=1}^{s_y} \sum_{\varphi=1}^{s_\varphi} x_{xy\varphi} \tag{I.1}$$

$$\sum_{\varphi'=1}^{s_\varphi} \sum_{x'=0}^{s_x} \sum_{y'=0}^{s_y} x_{x'y'\varphi'} \cdot c(x', y', \varphi', x, y) \quad \geq 1 \qquad \forall\, 1 \leq x \leq s_x, 1 \leq y \leq x_y \tag{I.2}$$

Work by Erdem and Sclaroff goes further along the lines of BIP solving and approaches the problem from a more AGP-like angle. The surveillance area is modelled as a convex polygon with convex polygonal holes (obstacles), and a section of the paper is dedicated to visibility analysis algorithms applicable in such a setting. Taking into considerations the need of higher-level image processing tasks, Erdem and Sclaroff suggest the inclusion of visibility or resolution constraints into the visibility algorithms. This creates a distinction between a camera's original visibility polygon, as it would be considered for the AGP, and a camera's feasibly visible region: the set of grid cells for which it can ensure sufficient image quality. To apply this additional restriction, the authors reduce the visibility polygon to points within a certain radius, effectively using partially occluded circular sectors as a field of view representation. The resulting visibility matrix is then used to construct the binary linear program, which is solved in MATLAB [133]. Experiments were run with a resolution of 100x100 grid cells, on floor plans including an exhibition hall and a parking lot. It is inferred that while a denser sampling of space and camera parameters would pull the solution towards the continuous optimum, it is often unnecessary to sample extensively to achieve satisfying near-optimality.

Giving up optimality, several papers can be mentioned which use heuristic approaches to solve the problem. Angella, Reithler, and Gallesio [5] introduce a branching greedy algorithm which maintains a memory of solutions near its construction path: whenever a tie has to be broken, the greedy registers both solutions so that it may eventually explore them both. The approach helps the algorithm escape from local optima by exploring several greedy routes in that fashion. For their camera orientation problem [144], Munishwar and Abu-Ghazaleh suggest a greedy approach over the set of all possible pans, and the availability of known camera locations allows for a hierarchical approach. Cameras are clustered based on their position, and a cluster leader solves the panning problem in its cluster before broadcasting its local solution. Another paper [143] from the same authors adds a distributed approach to the problem, for which clusters are assigned priorities to avoid overlaps between local solutions: the overlapping cover is allocated to the cluster with higher priority.

In their approach to 3D space sampling [4], Andersen and Tirthapura compare the results of a basic CPLEX [97] configuration for Branch-and-Bound, an adapted randomised greedy and ITEG, a greedy-based improvement heuristic introduced in [132]. In an attempt to compromise between network cost and coverage quality, Rebai et al. formulated the problem as a multiobjective binary linear program in [163], and combined the objectives into one through three approaches. The first, the weighted-sum method, simply aggregates the two contradictory objectives into one linear expression, with weights λ and $1 - \lambda$. Another solution is the two-phase method, based on the adaptive exploration of a Pareto front between the two objectives. Finally, the ϵ-constraint method converts one of the objectives into a constraint, and can be likened to a reverse Lagrangian relaxation [73]. All three methods were tested using a default CPLEX configuration [97]. Other papers using a greedy algorithm for either solving or initialisation include [78], and a 2013 review by Zhao et al. on BIP formulations and approximation methods can be found in [200].

Recent literature includes work by Yaagoubi et al. [189], who proposed a Voronoi segmentation of the surveillance areas using the buildings as cell generators, based on work by Dong [56]. A deterministic algorithm is applied which places cameras uniformly along the Voronoi edges such that they may cover the building entrances. The authors use omnidirectional cameras and focus on covering building walls and the roads in-between buildings, which approximately lay on the Voronoi edges.

For works which provided details on both environment modelling and solving approaches, a summary of this section can be found in Table I.1. The "space" column refers to the dimensionality of the surveillance area, while the "viewsheds" column indicates which camera representation model is used in the paper. The "occlusion" column holds a checkmark when the paper considers this additional constraint when performing visibility analysis computations. It is worth mentioning at this stage that due to the high diversity of applications and variants of the OCP, there is, to this day, no established benchmark to perform a fair numerical comparison of the algorithms applied to it. In this section, we therefore focus on design-related aspects but do not attempt to compare propositions based on their efficiency, as we feel this would inevitably create a bias.

Metaheuristic approaches

Given the $\mathcal{NP}$-hardness of the problem, the use of exact, optimality-oriented methods, becomes prohibitive as dimensionality increases. For this reason, researchers have come up with a variety of metaheuristic algorithms which aim at providing good approximations in a budgeted amount of time.

Ref.	Type	Space	Viewsheds	Occlusion
[63]	Single-objective BIP	2D	Circular sectors	✓
[95]	Single-objective BIP	2D	Triangles	✗
[5]	Greedy (branching)	3D	Depth maps	✓
[144, 143]	Greedy (distributed)	2D	Circular sectors	✗
[163]	Multi-objective BIP	3D	Cones	✗
[189]	Voronoi segmentation	3D	Disks (projection)	✓

Table I.1: Deterministic solving methods for the optimal camera placement problem

In the branch of evolutionary methods, genetic algorithms (GAs) [93] have been regularly tested on that problem. David, Idasiak, and Kratz [53] compare their implementation to an exact Branch-and-Bound algorithm. Their objective function uses bonuses for coverage and penalties for sensor setup, therefore relaxing the full coverage constraint. A parameter, the gain, defines how much more coverage a camera must provide to the current solution in order to be selected by the genetic operators. With a concern for backup coverage, Kim, Murray, and Xiao [111] used a GA to tackle the problem from a multiobjective angle and build a Pareto front of metaheuristic solutions. The algorithm is based on elitism, through which only non-dominated solutions can be carried into the next generation, and infeasibility is tackled by adjusting the chromosome structure before crossover. The results are two pools of individuals, one for each objective, defining the Pareto front. Similar work with GAs by Indu et al. [98] can be noted, although the authors focus on maximising both primary and overlapping coverage. With their game-based model, Fantini and Chaimowicz [68] also used GAs when the instance size exhausted their exact algorithm. In this case, the fitness function used to evaluate individuals focused on the number of watchmen and the ground samples they collectively covered.

Similar in their nature-inspired aspect, particle swarm optimisation (PSO) algorithms [107] have also displayed some popularity. Early work includes Conci and Lizzi [41] for coverage with essential regions, although very few details are given about the algorithm. In two papers [140, 141], Morsly, Djouadi, and Aouf focus on PSO for their best interceptor strategy, and for camera placement in general. The earlier paper introduces probability-inspired binary particle swarm optimisation for this problem, and compares it favourably to two PSO variants: improved [80] and novel [108], along with other metaheuristics. The method improves on the standard binary PSO algorithm by computing particle velocities based on each bit's likelihood to be 1. This probability is based on the bit's value in the current best global solu-

tion. More recently, this approach has been extended by Fu, Zhou, and Deng [77], who used a different update operator once the new velocities have been computed. PSO can also be found in work by Konda and Conci [115], which remains in the continuous domain for camera parameters, but evaluates a particle's fitness using the sampled floor plan and a ray tracing algorithm. Further experiments with PSO were conducted in [188] by Xu, Lei, and Hendriks. The paper compares three ways of handling violating dimensions in infeasible solutions: absorbing (reset to its initial value), reflecting (set to its opposite value), penalising (strong fitness penalty for infeasibility).

Artificial bee colonies (ABCs) [104] and artificial spiders have also been tested on the problem, with two publications [37, 38] by Chrysostomou and Gasteratos. In this work, additional visibility constraints are formulated (resolution, viewing distance, occlusion) and aggregated into the fitness function for evaluation. The algorithm then proceeds as usual, identifying elite sites for neighbourhood exploration (local search) and dispatching the remaining bees randomly (global search). The authors later provided summarised results for both coverage maximisation and cost minimisation [37], however the absence of more numerical data renders the comparison with other algorithms rather unfair. An extension involving so-called artificial spiders to the process is also attributed to the same authors [38]. In this paper, artificial bees are used to determine the optimal number of cameras, while the spiders algorithm computes their actual positions and orientations. Unfortunately, the inner workings of this newly added bio-inspired method are not thoroughly defined. Table I.2 brings together the aforementioned nature-inspired methods.

Ref.	**Type**	**Space**	**Viewsheds**	**Occlusion**
[53]	GA	3D	Depth maps	✓
[111]	GA	–	–	–
[98]	GA	3D	Circular sectors	✗
[41]	PSO	2D	Circular sectors	✓
[140]	PSO	2D	Triangles	✗
[141]	PSO	2D	Pyramids (projection)	✗
[68]	GA	3D	Human eyesight	✓
[115]	PSO	3D	Pyramids	✓
[188]	PSO	2D	Circular sectors	✗
[77]	PSO	2D	Pyramids (projection)	✗
[37, 38]	ABC	3D	Pyramids	✓

Table I.2: Nature-inspired approaches to the optimal camera placement problem

Other metaheuristics for optimal camera placement (Table I.3) have included simulated annealing (SA) [114] and its transdimensional variant (TDSA) [26], differential evolution (DE) [177], and custom local search algorithms. Simulated annealing has been used by Janoos et al. [101] in order to help the Nelder–Mead method [150], used for solving, escape the local optima it tends to easily fall into. Transdimensional simulated annealing has been explored by Liu et al. [126, 127] for the same problem. The idea is to consider a stochastic optimisation framework which allows SA to evaluate both changes in parameter values (moving a camera) and in dimensionality (adding or removing a camera). Attempts to use differential evolution for optimal camera placement are still very recent, with first one work [195] by Zhang et al. The metaheuristic requires the same adjustment as PSO to tackle combinatorial problems, and behaves similarly to genetic algorithms, save that the crossover operator is based on a problem-specific difference function between individuals rather than on a selective union of their properties. In their paper, Zhang et al. represent an individual as a vector of camera indices, and the following crossover operator is iteratively applied. For every individual in the population, two others are randomly selected, and their element-wise difference is computed. The result is then scaled down using a fixed zoom factor, and added to the first individual to create a so-called mutant. Then, when constructing the next generation, each gene of the original individual is replaced with the corresponding one in the mutant, with given crossover probability. A small bias in the algorithm ensures that at least one mutant bit is carried over. More recent work involving DE for the OCP can also be found in [25] by Brévilliers et al. While the paper focuses on preprocessing distribution and parallelisability, it also includes a hybrid set-based DE algorithm inspired from work by Maravilha, Ramírez, and Campelo [130]. In their implementation, mutant individuals are generated by combining a random solution with two others taken from the population. Once set operators have been applied, a mutant is defined which consists only of cameras registered in the aforementioned solutions. A smaller instance, restricted to this mutant's cameras, is then solved exactly using CPLEX [97], which effectively serves as a crossover operator. The resulting solution can then be evaluated for selection into the next generation.

For the sake of completeness, the following papers may also be noted which introduce other search procedures for the problem. In [141], Morsly et al. include a tabu search (TS) algorithm [84, 85], along with generic GA and SA implementations for comparison with their own PSO work. Fantini and Chaimowicz define a backtracking exhaustive search algorithm in [68], and use it whenever the instance size permits it. Finally, for their work in generic sensors network design [164], Rebai et al. propose a local search procedure, the neighbourhood for which is defined by

adding a so far unused sensor location to the graph, and removing existing elements which were made redundant by the new node.

Ref.	Type	Space	Viewsheds	Occlusion
[101]	Nelder–Mead/SA	3D	OpenGL	✓
[68]	Backtracking	3D	Human eyesight	✓
[127]	TDSA	2D	Disks	✓
[195]	DE	3D	Pyramids	✓
[25]	DE	3D	Pyramids	✗

Table I.3: Other metaheuristics for the optimal camera placement problem

I.3 The set covering problem

Definition and relationship to camera placement

Most of the specifics for the optimal camera placement problem are usually treated in the visibility analysis phase, during which the instance translates into a visibility matrix from each camera to each point in the discretised space. As an example, consider the Figure I.3 floor plan used in Section I.2. The sampling process for this case defines an OCP instance with 37 ground samples. Using the walls around and inside the building, 60 possible camera locations can be selected. For the sake of simplicity, we will assume that the orientation sampling process ran on a 2π range for all positions. Given a $\frac{\pi}{6}$ pan sampling frequency (12 possible pan angles) and one possible tilt angle, the 60 aforementioned positions correspond to $60 * 12 * 1 = 720$ possible camera candidates for the optimisation process to choose from (see Figure I.5b).

Before this can happen however, the ground samples and camera candidates need to be matched. In other words, the optimisation algorithm needs to know which candidates can cover which points. This stage was discussed in more details in Section I.2, and typically yields a visibility graph of which Figure I.5a shows a subgraph. From this perspective, the reader may recognise the popular dominating set graph theory problem, whose combinatorial equivalent is one of Karp's well-known $\mathcal{NP}$-hard problems: set cover [105]. In a generic manner, it is usually formulated as follows:

(a) The frustum for one candidate and its associated visibility subgraph

(b) All possible frustums

Figure I.5: Bottom-right corner of the Figure I.3 floor plan with the associated camera candidates. The horizontal field-of-view was set to 65° and the range to 2 units

Problem (set cover)**.** *Given a set of elements $\mathcal{I}$ (rows) to be covered, and a collection of sets $\mathcal{J}$ (columns) such that the union of all sets in $\mathcal{J}$ is $\mathcal{I}$, find the smallest subset $\mathcal{C} \subset \mathcal{J}$ such that $\bigcup_{e \in \mathcal{C}} e = \mathcal{I}$. In other words, determine the smallest subset of $\mathcal{J}$ which covers $\mathcal{I}$. By assigning a cost to each column, the problem can also be that of finding the subset with minimal cost.*

For convenience, we will also be using the notations $\mathcal{I}_j$ for the set of rows covered by column j, and $\mathcal{J}_i$ for that of the columns which cover i. Letting $\mathcal{I}$ be the set of all coverable ground samples in the surveillance area, and $\mathcal{J}$ the covers of all camera candidates, expressed as subsets of $\mathcal{I}$ determined through visibility analysis (see Section I.2 and Figure I.5), the optimal camera placement problem is structurally identical to set cover, and is therefore $\mathcal{NP}$-hard.

In spite of this relationship, to the best of our knowledge, most set cover literature has yet to find its way into work done on optimal camera placement. Furthermore, there appears to have been no published works which bring knowledge acquired from the OCP into SCP literature. Given that the former is a specialisation of the latter, the reader might argue that this comes as no surprise. It is also worth noting that any extension of the surveillance area in an OCP instance, or worse, the inclusion of an additional dimension, generates a significant growth of $\mathcal{I}$, making transformations to set cover highly sensitive to dimension changes in the original camera placement problem.

Instance reduction

To lighten the curse of dimensionality for the SCP, two main instance reduction procedures were defined by Beasley [12]. The first, column domination, identifies dominated columns—that is, columns whose covers are subsets of others. More formally, the set of removable dominated columns is given by:

$$\mathcal{D} = \{j \in \mathcal{J} \mid \exists j' \in \mathcal{J}, j \subset j'\}$$

It is worth noting that this pruning procedure is more straightforward on unicost instances, since weighted variants would have to take cost into consideration [165]. The second method, column inclusion, is a look-ahead procedure identifying columns which have a monopoly over some rows. In that case, to ensure coverage, the columns are immediately added to the solution and removed from the instance before solving. More formally, the set of included columns is given by:

$$\mathcal{I} = \{j \in \mathcal{J} \mid \bigcup_{\substack{j' \in \mathcal{J} \\ j' \neq j}} j' \neq \mathcal{I}\}$$

These procedures have found their way into many pieces of recent set cover optimisation literature, including but not limited to [123, 165, 51]. They have been used and extended in this thesis, although the second procedure (inclusion) was found to not be suitable for human-assisted optimisation, as will be shown in Chapter IV.

Greedy and heuristic algorithms

As a well-known problem, set cover has obviously attracted a lot of attention from research in deterministic methods, even though its $\mathcal{NP}$-hardness makes the use of exact methods highly impractical. Pioneering work on the problem is usually attributed to Chvátal who designed a greedy procedure [40] for the problem, and proved that it would never overshoot the optimal solution by more than a factor of $\sum_{i=1}^{d} \frac{1}{i}$ with $d = \max_{e \in \mathcal{J}} |e|$, the maximal cardinality in $\mathcal{J}$. A tighter bound was later found by Parekh [157], and more recently, Felici et al. pointed out an a-priori upper bound using probabilistic methods [71]. Since these early papers, the development of better greedy algorithms for set cover has remained a popular line of research, and many algorithms, both deterministic and stochastic, still use them to build or repair solutions. A recent paper by Chandu [33] reports on tests involving a generic set-based greedy, which includes several columns at once. Ablanedo-Rosas and Rego [1] also suggested the use of surrogate models for problem relaxation and compared several

update rules including Chvátal's. A 2016 review of various greedy heuristics was published by Vasko, Lu, and Zyma in [186].

As a note on the connection between our two problems, it is worth noting that OCP literature has in fact included the use of greedy algorithms, as we have seen in Section I.2. While the algorithms mentioned there were typically tailored for the OCP, the key ideas remain quite similar. In fact, when considering Chvátal's selection operator—which maximises coverage at every iteration—one can draw a parallel with the work of Angella, Reithler, and Gallesio [5] and Munishwar and Abu-Ghazaleh [144, 143].

In any case, although greedies are very attractive in that they quickly provide acceptable primal bounds for other heuristics on which to build, the usual Branch-and-Bound scheme remains the only way to explore the search space exhaustively and find a provably optimal solution. While the usual approach—which uses the problem's linear relaxation, the simplex algorithm and variable-based branching—has remained prohibitive for $\mathcal{NP}$-hard problems, several observations on the SCP have allowed the implementation to be customised so as to speed up gap reduction. Early success along those lines must be credited to Etcheberry [66] who suggested a new branching strategy and the use of Lagrangian [73] rather than linear relaxation at every node. The enumeration algorithm uses subgradient optimisation to acquire better Lagrange multipliers and a possibly better lower bound at every step. Balas and Ho later designed a primal-dual enumeration algorithm [9] using dual and greedy primal heuristics, as well as cutting planes and variable fixing for gradual instance reduction. The branching scheme is taken from [66], and Beasley extended the work with a three-phase algorithm for dual bounds in [12]. This paper also elaborated on the previous instance reduction procedure, which uses the latest Lagrangian costs and dual bound to eliminate columns at every iteration. Further work on this approach can be found in [13, 15, 8, 29, 191, 32] and a review entitled "Algorithms for the Set Covering Problem" by Caprara, Toth, and Fischetti can be found in [30].

Moving on to other approaches, a paper [131] by Marchiori and Steenbeek introduces an elitist algorithm which gradually reduces the problem by maintaining a core of elite columns. At every iteration a randomised greedy is restarted from the most frequently selected columns so far, and the score of every column in the resulting cover increases. An optimisation routine then prunes the solution using a pairwise variant of the column domination algorithm introduced earlier. In a similar attempt to avoid local optima, Lan, DePuy, and Whitehouse introduced a non-deterministic greedy algorithm [123] which sometimes selects columns within an acceptable range of the greedy choice. A neighbourhood search heuristic is then used to improve the resulting solution, and the overall proposition has proven to be quite effective.

With a focus on the more difficult unicost variant of the SCP, Gao et al. propose a row-weighting local search algorithm (RWLS) in [79]. In this approach, every row is first assigned a weight, initialised at 1. Every column which is not in the current solution is then given a score corresponding to the sum of its uncovered rows' weights. Columns in the current cover are scored negatively, by summing up the weights of the rows they are the only ones to cover, and assigning the opposite values. Given these attributes, a local search algorithm is then run which first breaks feasibility by removing selected columns with high, albeit negative scores. An additional column is then removed, and a uncovered row is chosen randomly. The highest-scoring column for this row is added to the solution, and the weights of the remaining uncovered rows increased. A tabu list is used to prevent loops, and timestamps help break ties by ensuring that recently selected columns cannot be reused immediately. As will be confirmed in Chapter III, this proposition by Gao et al. is among the most promising approaches for both the unicost SCP and the OCP.

Nature-inspired methods

Although some of the algorithms introduced earlier can reach thousands of rows and columns, the sampling processes used for optimal camera placement can generate SCP instances which can reach as far as or beyond the challenges of the standard OR-Library [13] (typically used for SCP benchmarking), even for simplistic surveillance areas. Acknowledging the existence of such large instances in practice, the set cover community has given a significant amount of time to metaheuristics, which effectively sacrifice optimality in order to yield acceptable solutions in a limited amount of time.

Among those, the greedy randomised adaptive search procedure (GRASP) was introduced using the SCP Feo and Resende [72] and effectively removes the determinism in Chvátal's algorithm. Instead of always choosing the column with the most uncovered elements in its cover, the algorithm picks one randomly among those within a given factor of that maximum gain. To benefit from randomness, the procedure is repeated several times, and the best solution across all iterations is returned with its redundant columns removed. Following up on GRASPs, Bautista and Pereira derive a specialised routine in [11], inspired from efficient SAT-solving heuristics, for the reputably harder to solve unicost SCP. The instance is reduced to SAT, the original $\mathcal{NP}$-complete problem [42], for which many heuristics and solvers have been designed.

Another early and influential paper to tackle the SCP metaheuristically was published by Beasley and Chu in 1996 [14]. The authors introduced an efficient genetic algorithm for set cover which, to this day, remains a good reference for comparison.

The selection operator is tournament-based: the population is randomly split in two groups, and the best of each are matched iteratively until both groups are empty. The crossover operator is fitness-biased, meaning that the value of each of the child's bits is more likely to come from the parent with better fitness. The mutation operator flips each bit with some probability, and the authors decided to dynamically adjust the mutation rate as the algorithm converges. Infeasible solutions are repaired using a greedy heuristic, duplicate solutions are ignored, and a steady-state replacement policy is used for population updating: new solutions replace below-average individuals. This implements elitism, in that above-average individuals cannot be replaced. A 1997 review [128] of GAs for the SCP was later published by Lorena and de Souza Lopes which highlights the high performance of even a classical GA implementation on this problem.

Later work includes [3] by Aickelin, which tackles the problem less directly. Rather than to attempt to solve the problem straightforwardly by manipulating columns, the algorithm aims at finding the best ordering of rows, using a so-called decoder as a fitness function. Once an ordering has been generated within the GA's population, it is deterministically decoded into an SCP solution by a greedy algorithm which iterates over the rows and for each one selects the most interesting column. The greedy's parameters, which define a column's attractiveness, are also part of the GA's individuals and are optimised along with the ordering.

A more straightforward GA implementation has also been attempted for the SCP by Kornilakis and Stamatopoulos in [117], with an application to airline crew pairing. The objective function uses application-specific penalties, while the other components are typical of the usual GAs. The selection operator uses a fitness-biased roulette wheel, the crossover operator is uniform, meaning genes may come from either parents with equal probability. The mutation operator is less standard: a fixed number of genes is selected in the individual, and each of them is flipped with a probability based on the density of zeros in the population's fittest individual.

Solar, Parada, and Urrutia also extend regular GAs in [174] by introducing a parallel implementation of the algorithm. The authors generate several populations, running their own GAs independently and waiting for each other at the end of each generation. Then, each algorithm sends its best individual to a designated master node, which picks the best among them and broadcasts it back to all other nodes. Individuals follow a binary representation, and infeasible solutions are repaired greedily. Another, more recent attempt involving parallel genetic algorithms can be found in [197] by Zhang et al.

An example which tackles the infeasibility cases emerging during crossover can be found in [64] by Eremeev. Rather than to use an indicator vector mapping $\mathcal{J} \rightarrow$

$\{0, 1\}$, a chromosome is made up of $|\mathcal{I}|$ genes, the values of which are indices of $\mathcal{J}$. A gene therefore represents a row, and its value corresponds to a column which covers it. Once two parents have been randomly selected with the usual fitness bias, the union of their columns is used to define a small SCP subproblem, which can be reduced and solved. Given a valid initial population, in which all genes reference columns actually covering their rows, and considering the above crossover operator, one can see that the algorithm will indeed never generate infeasible solutions. The mutation operator does not compromise this property, and simply changes a row's column to another which covers it, with given probability for each gene. It is worth noting that the infeasibility problem tackled here has been the subject of more recent work, with Bilal, Galinier, and Guibault proposing an unconstrained set cover formulation in [18], based on gains and penalties in the objective function.

Acknowledging the efficiency of genetic algorithms on the SCP, Crawford et al. proposed a variant called cultural algorithms [46, 44] which adds longer-term memory into the original GA design. The authors defined a so-called belief space which, across generations, retains two individuals, used during crossover: best fitness and most diverse so far. The belief space is updated at the end of every generation if a new individual acquires a better value for one of these two metrics. The mutation operator and the repair function are taken from Beasley and Chu [14].

Genetic algorithms made up the first references in OCP Section I.2 earlier. The constraint relaxation approach found in both Bilal, Galinier, and Guibault [18] and David, Idasiak, and Kratz [53] is one of very few elements found in both bodies of literature, in spite of the approach's popularity in SCP papers. The reader will note that this relates to the weighted SCP, which may explain this discrepancy.

Ant colony optimisation (ACO) algorithms [57] have been applied to the SCP, with early work by Lessing, Dumitrescu, and Stützle in [124]. The paper highlights the importance of good initial heuristic information, which is used before the pheromones trails are sufficiently strong, and compares several approaches. In their work, Lessing, Dumitrescu, and Stützle use these seven sources of heuristic data and compare them using three ACO algorithms: the original ant colony system [57] by Dorigo, Maniezzo, and Colorni, the $\mathcal{MAX}$-$\mathcal{MIN}$ ant system [178] by Stützle and Hoos, and a hybrid variant combining the previous two [179]. The approximate nondeterministic tree-search (ANTS) algorithm [129] by Maniezzo is also included for comparison, and a local r-flip search is used for neighbourhood exploration (within r column changes). This local search algorithm, which has been applied standalone on the SCP, was brought forward by Yagiura, Kishida, and Ibaraki in [191]. Lessing, Dumitrescu, and Stützle concluded that without local search, cover gain was the most interesting source of heuristic data. When a local search procedure was added,

Lagrangian lower bounds proved more efficient.

Later work with ACOs has included [43] by Crawford and Castro, who suggested the use of constraint programming and restricted arc consistency to detect infeasible partial solutions ahead of time, as well as solution postprocessing as an alternative to local search. Inspired by the original ACO results on the travelling salesman problem, Ren et al. proposed another approach [165] and tackled the problem in a row-oriented fashion. At every iteration, the ants select uncovered rows at random and columns are chosen, using the ant colony's information, among those which can cover them. A similar approach which appears to be less dependent on local search improvement can be found in [142] by Mulati and Constantino. Valenzuela et al. [185] later referenced both Aickelin [3] and Crawford and Castro [43], and proposed a hybrid method involving ACO, scatter search [83], and a genetic algorithm to tune the parameters of the previous two. The latter uses fitness-biased random selection, single-point crossover and random uniform mutation. The ant colony follows its original implementation. More recently, Al-Shihabi, Arafeh, and Barghash proposed the use of linear relaxations for initial problem reduction, and adjusted the implementation of the $\mathcal{MAX}$-$\mathcal{MIN}$ ant system for the SCP [172]. To this day and to the best of our knowledge, ACOs have yet to yield interesting results on the unicost SCP or on the OCP.

As far as nature-inspired methods are concerned, two others popular metaheuristics are also worth mentioning here. Crawford et al. suggest the use of an artificial bee colony for the SCP in [48]. The population is initialised by selecting a random column for each row, and worker bees adjust their position by adding and removing columns based on a randomly selected food source. Infeasible solutions are repaired, presumably greedily, although the paper does not provide more information. The results seem to suggest that ABCs are particularly efficient on large-scale SCP instances. Earlier work had included [180] by Sundar and Singh which combined the method with a local search procedure, with promising results. More recently, Balaji and Revathi [7] proposed PSO as an approach for the SCP. The authors used a variant [81] introduced by Garcia and Perez, inspired from frogs rather than birds, and designed for combinatorial problems. Similar inspiration can be found in [51], in which Crawford et al. apply the shuffled frog leaping algorithm, originally formulated by Eusuff and Lansey in [67], to the SCP. The algorithm was adjusted for binary-encoded individuals through the use of transformation functions originally studied by Mirjalili and Lewis in [135]. Other metaheuristics inspired from nature include [103] by Joshi, Rowe, and Zarges who exploited recent advances on our understanding of the human immune system to develop a parallel germinal centre artificial immune system algorithm. Work along the same lines can be found in [184] by Tasnim, Rouf,

and Rahman.

As the reader may recall from Section I.2, ABCs have not been tremendously popular in OCP literature (see work by Chrysostomou and Gasteratos [37, 38]). PSO, on the other hand, ranks among the most popular approaches. The designs however remain very different as far as their key features are concerned: SCP literature seems to bring forward the idea of different neighbourhoods or clusters used in the update equations, while OCP papers stray further away from the continuous versions of the algorithm.

Among other approaches, simulated annealing has also been applied to the SCP, with a 1995 publication [100] by Jacobs and Brusco. For this algorithm, the initial solution is built by iteratively selecting a random row, and including the first column to cover it into the solution. Redundant columns are then removed, and the cooling process begins. Neighbourhood search is performed by first dropping a fixed proportion of columns from the solution, and repairing it by adding columns the costs of which are within a given factor of the maximum column cost in the instance. The algorithm was tested for various parameter values, and improved many of the best known solutions at the time. Brusco, Jacobs, and Thompson later explored the idea again in [27], for instances with strong cost-to-coverage correlation and using a so-called morphing procedure for local search on partial solutions. For another single-solution approach, Musliu define a local-search procedure [146] using a move-oriented neighbourhood, which allows for the easy use of a tabu list to avoid cycles. The authors define a constructive and a destructive move, and set rules on their usability at each given stage of the algorithm.

Finally, being a popular problem, the SCP has also led to the creation or testing of recent algorithms, inspired from concepts mostly unexplored in the field of metaheuristics. These include physics ([147, 6, 166, 175]), chemistry ([194]), wildlife ([49, 45, 47]), pyrotechnics ([50]), music ([168]) and sports ([102]), among others. The list of all such propositions is quite extensive: we therefore refer the reader to [119] for complete details and numerical comparisons.

Review

While we were able to establish some connections between OCP and SCP literature, this relationship is clearly under-exploited. Very little of the knowledge we have about the set covering problem has found its way into camera placement literature, or vice-versa. As will be shown in Chapter II, OCP instances do tend to have specific characteristics which call for tailored algorithms. Nonetheless, a better

understanding of the leading algorithms for set cover does provide valuable insight. As a summary, we provide a list of said algorithms in Table I.4, focusing on the weighted set cover problem. As far as the unweighted variant is concerned, the state of the art is much simpler. The RWLS algorithm mentioned earlier remains mostly unchallenged, although very recent work seems to suggest hybridising approaches might be promising [159].

Year	**Ref.**	**Type**	**Proposition**
1999	[29]	Heuristic	Lagrangian-based framework
2000	[131]	Metaheuristic	Elitist evolutionary algorithm
2006	[191]	Heuristic	3-flip neighbourhood search
2007	[123]	Heuristic	Non-deterministic greedy algorithm
2010	[6]	Metaheuristic	Gravity-inspired search algorithm
2010	[180]	Metaheuristic	Artificial bee colony with local search
2014	[194]	Metaheuristic	Chemical reaction optimisation
2015	[172]	Metaheuristic	$\mathcal{MAX}$-$\mathcal{MIN}$ ant colony optimisation

Table I.4: The 8 algorithms which reached 100% BKS on the 1987 and 1990 OR-Library instances

In this thesis, several elements will be taken from set cover literature and integrated into our approach. This includes extended reduction procedures, weighting schemes, greedy heuristics as well as another, closely-related problem variant which we introduce in Chapter IV.

Chapter II

Problem formulation and instance generation

Introduction

Having reviewed the wide variety of variants, constraints, objectives and applications which surround the optimal camera placement problem, we now establish the modelling basis for this thesis. In this chapter, we begin by reintroducing a basic set cover, full coverage formulation of our problem. This model, combined with feedback from other research partners, then serves as a basis for the design of an instance generation procedure which brings optimal camera placement into the real world. We introduce tailored sampling procedures which capture knowledge from the layout of the given area, using real world, publicly available data. The integration and extension of reduction procedures from set cover literature is also discussed and a first solve using three basics algorithms yields interesting hypotheses to be studied in Chapter III and used in Chapter IV.

Related publication

[120] Julien Kritter et al. "On the real-world applicability of state-of-the-art algorithms for the optimal camera placement problem". In: *6th 2019 International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, Apr. 2019. DOI: 10.1109/CoDIT.2019.8820295

II.1 Problem formulation and generation overview

In order to first focus on real-world data processing, we begin with a very simple formulation, which follows that of the set cover problem or early camera placement literature [63, 95]. We therefore work from a set of ground samples $\mathcal{I}$, defined as points in a 3D cartesian coordinate system, and a set of camera candidates $\mathcal{J}$ for which we add orientation information in the form of a normalised 3D vector. A visibility matrix A with elements a_{ij} is also available and maps sets $\mathcal{I}$ and $\mathcal{J}$ together. The problem is therefore given by the following binary integer program, in which decision variables x_j determine whether or not a candidate is selected.

$$\textbf{min} \quad \sum_{j=1}^{|\mathcal{J}|} x_j \tag{II.1}$$

$$\sum_{j=1}^{|\mathcal{J}|} a_{ij} x_j \quad \geq 1 \qquad \forall i, 1 \leq i \leq |\mathcal{I}| \tag{II.2}$$

This formulation is of course that of the unicost set cover problem, and enforces **full coverage** across the entire surveillance area while minimising cost.

Before such a model can be used however, a procedure needs to be designed to acquire $\mathcal{I}$, $\mathcal{J}$ and elements a_{ij} of the visibility matrix. Several approaches have been put forward by the authors reviewed in Chapter I, however those approaches tend to be either overly simplistic or very computationally expensive. In this thesis, we propose a middle ground between those two extremes, which captures information from the real world while mostly keeping algorithm complexity down. To this end, our approach uses data sources. The first is the OpenStreetMap (OSM) database [155].

> OpenStreetMap is a free, editable map of the whole world that is being built by volunteers largely from scratch and released with an open-content license.
>
> (The OpenStreetMap wiki: about OpenStreetMap)

OSM is a very complete, user-contributed world map (Figure II.1) which not only provides layout information such as roads or buildings but also an extensive tagging system which stores a significant amount of metadata for each element. In

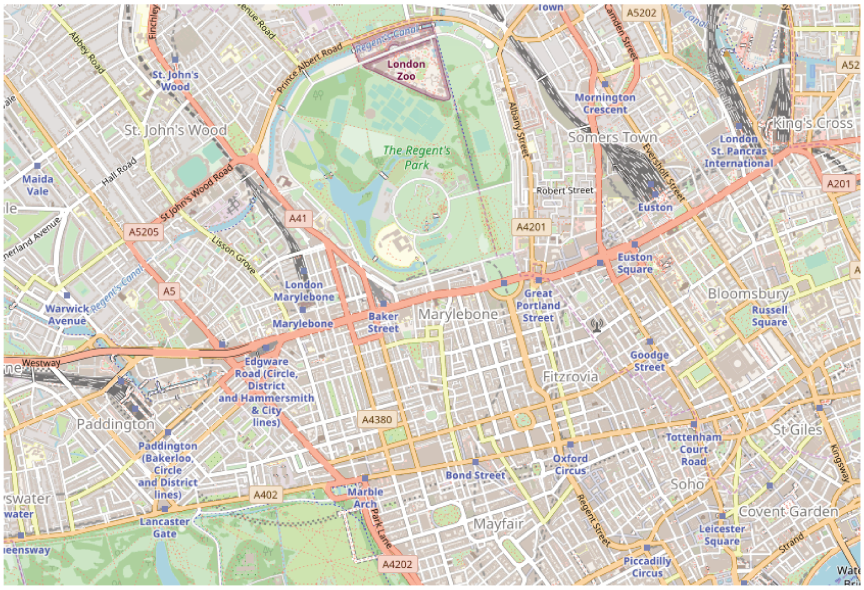


Figure II.1: The OpenStreetMap web viewer

this thesis, we use OSM to set the bounds of our surveillance area and generate both ground samples and camera candidates.

One drawback of OSM however is that, as a map, its data is two-dimensional. While building dimensions are available through the tagging system, ground elevation remains unknown and our instances remain flat. To solve this issue, we introduce another data source: NASA's Shuttle Radar Topography Mission (SRTM) data [176]. With over 80% of the Earth covered, the SRTM provides ground elevation data for every 1-degree longitude, 1-degree latitude square on the planet. At 1 arc second resolution, this represents an elevation value in meters for every 30 meters square.

With this data available, our instance generation procedure is as follows. First, OSM entities which are relevant to our problem are parsed and sorted. The geographic coordinates of each element are then transformed into a local three-dimensional cartesian coordinate system, which greatly simplifies our computations while remaining very accurate for our use cases. Elevation is then brought in, and the sampling process can begin. Ground elements such as roads and open areas are processed to yield ground samples, while buildings and walls provide camera candidates. Every pair is then passed on to the visibility analysis and instance reduction algorithms before the instance can be stored on disk.

II.2 Sampling procedures

OpenStreetMap entities and coordinate system transform

OpenStreetMap data revolves around four main entities: **nodes**, **ways**, **relations** and **tags**. Nodes are points in space given by a latitude and a longitude coordinates. They may represent elements of interest on their own such as traffic lights or electric poles, but are mostly used to build ways. These are sequences of nodes which can form lines or polygons to represent roads, buildings, open areas and various sorts of infrastructure. Relations do not correspond to physical elements but rather serve as a way to associate nodes and ways together. It is therefore common to find relations which group several building ways to represent them as parts of institutions or administrative regions. Finally, tags are key-value pairs which can be associated with any of the aforementioned entities. They provide extensive metadata about the nature and description of every feature on the map. They allow us, for example, to determine whether a polygon way is a building or a park.

Before we use this information to classify and process entities, we first need to move into a more convenient coordinate system. While we could, indeed, work in the geographic system from beginning to end, this seems like an unnecessary struggle given that our transform will only incur a minor precision loss, provided that our surveillance areas do not span across extremely large regions. This can reasonably be assumed since, in practice, CCTV infrastructure is designed locally and hardly ever goes beyond the scale of a city or conurbation. At this level, the loss of accuracy remains barely perceptible.

Our transform requires two stages. The first will convert the geographic coordinates (latitude, longitude) into Earth-Centred, Earth-Fixed (ECEF) geocentric cartesian coordinates. In other words, our nodes are repositioned into a system which has its origin at the Earth's centre. Its horizontal axes (x and y) are aligned with the prime meridian and the equator line, while the vertical (z) axis follows the direction of the true north. For a node at latitude ϕ and longitude λ, the transform is given by Equation (II.4). Equation (II.3) is an intermediary step for the computation of the prime vertical radius of curvature at latitude ϕ. This corresponds to the perpendicular distance from the surface at latitude ϕ to the polar axis [23]. It is computed using two parameters a and b which are the lengths of the Earth's major and minor semi-axes respectively. In Equation II.4, E represents the Earth's squared

eccentricity and is given by $\left(\frac{b}{a}\right)^2$.

$$n = \frac{a^2}{\sqrt{a^2 cos^2(\phi) + b^2 sin^2(\phi)}} \tag{II.3}$$

$$ecef_{\phi\lambda} = \begin{pmatrix} ncos(\phi)cos(\lambda) \\ ncos(\phi)sin(\lambda) \\ nEsin(\phi) \end{pmatrix} \tag{II.4}$$

Since we are working on small areas (relative to the Earth surface), the coordinates of our points in the ECEF system are bound to be very close. Since such a configuration does not exactly agree with a computer's approach to floating point numbers, our selected transform includes a second step to move from ECEF to ENU (East, North, Up) coordinates. This second system relies on the use of a local tangent plane. Given a reference point at (ϕ_0, λ_0), such a plane is defined by its normal vector, tangent to the Earth's surface at the reference. For our work, we have consistently chosen this point at the centre of the surveillance area's bounding box. The two other (orthogonal) axes are then set in the direction of the Earth's true north axis for one, and that of the east for the other. ECEF coordinates can then be translated onto this plane using Equation (II.7). Equation (II.5) computes the difference vector between a given node's ECEF coordinates (ϕ, λ) and that of the reference point. Equation (II.6) is included for notational convenience.

$$d = (d_x, d_y, d_z) = ecef_{\phi\lambda} - ecef_{\phi_0\lambda_0} \tag{II.5}$$

$$t = d_x cos(\lambda_0) + d_y sin(\lambda_0) \tag{II.6}$$

$$enu_{\phi\lambda} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -d_x sin(\lambda_0) + d_y cos(\lambda_0) \\ -tsin(\phi_0) + d_z cos(\phi_0) \\ tcos(\phi_0) + d_z sin(\phi_0) \end{pmatrix} \tag{II.7}$$

Note that by using the meter as the unit in the first step, we ensure that the ENU coordinates also follow this rule, save for the minor loss of precision incurred in the second step.

Now that each node is associated with cartesian coordinates local to our surveillance area, we are ready to integrate our elevation data. As was mentioned earlier, the SRTM data provides an elevation value in meters for every 30-meter square on Earth. A straightforward approach is therefore to add this value to the z coordinate of every node in each square. This however would generate a tiling effect on the floor, especially where the nodes are too densely packed together. To compensate for this, we have implemented smoothing into our elevation process. For a given node, we

first locate the associated SRTM tile in the dataset and determine in which quadrant of this tile our node lies. Based on this, we also fetch the two neighbouring tiles. For example, if our node is in the north-west quadrant of its tile, we fetch the northern and western neighbours. A weighted average elevation value is then computed based on the distances between our node and the centres of the three aforementioned tiles. This effectively transforms the original tiled landscape into a smooth curved surface which better represents the ground.

With the transform and elevation complete, we can now move up to the other OSM entities. It should be noted that while they are essential for the implementation, relations can effectively be ignored by conceptually transferring their tags over to their node or way members. In the rest of this section, we will therefore focus on nodes (points) and ways (polylines and polygons).

Using the tagging system and filtering rules dictated by the users and the study case, we separate nodes and ways into what well call ground and structural entities. Put simply, ground entities yield ground samples, that is, points in space that must be covered and which represent our continuous surveillance area. Structural entities yield camera candidates which can provide said coverage. Note that this topology will be reformulated in Chapter IV as we integrate more application-specific constraints into our model.

Ground entities

Ground entities are divided into three categories for processing. Ground points are single points of interest represented as nodes in the data. They typically include pedestrian crossings but may represent all sorts of elements. Only one location is of interest in this case: that of the node. Ground lines are OSM ways and represent various types of roads. Finally, ground polygons are used for open areas such as parks, parking lots, pedestrian areas, and so on.

Locations of interest along a ground line are computed using a simple 3D polyline walk. Starting at the first node, the algorithm advances along the line at a predetermined frequency and registers its location at every stop. When the sampling frequency does not exactly divide the way's length, the samples are centred by removing the excess margin at both ends. At every registered location so far, the algorithm also takes into account the road's width, which we extract from tags or default values set to represent regulatory lane widths. This information is used by walking orthogonally to the lane at a preset frequency and registering new locations. Two parameters are therefore required for ground line sampling: the ground length frequency f_g^l for the walk along the way and the ground width frequency f_g^w for the

orthogonal walks (both in meters). Their exact use is illustrated in Figure II.2: the darker samples follow the line using parameter f_g^l while the lighter ones surround them at distance f_g^w.

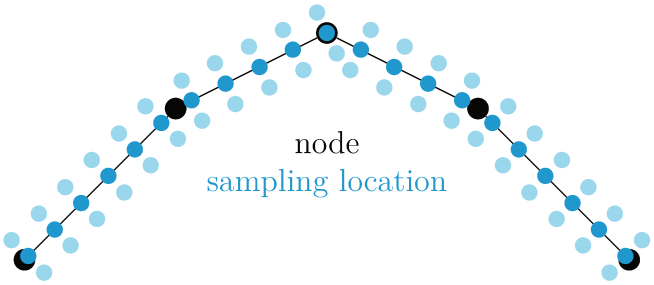


Figure II.2: Illustration for the ground line sampling procedure

Ground polygons require more steps, especially as there are no guarantees about their convexity. Various approaches have been tried to sample these geometries and we have selected the following, which we illustrate in Figure II.3. First, the polygon is triangulated using an ear-slicing algorithm. We used an implementation from Mapbox [59] which the authors reported to be inspired from [75, 58]. Each triangle is then processed individually. First, the vertex which connects the two longest edges is identified. Lines are then drawn parallel to the remaining edge at frequency f_g^l and locations of interest are registered by walking those lines at frequency f_g^w following the same process as for ground lines. When the triangle is too small for the length-wise walk, a single location is registered in its centre instead.

Now that locations of interest have been extracted from all available entities, the third dimension must be taken into consideration. To account for human height, additional locations are registered above the ones available so far, using three new parameters: a_g^{min} and a_g^{max} will bound the coverage altitude levels while f_g^a will determine the frequency at which new locations are to be registered.

All the coordinates computed so far serve as our set of ground samples $\mathcal{I}$. The sampling process has ensured that this set is a fair representation of the surveillance area's layout as it is described on the map. Of course, as the reader will no doubt notice, this process already requires a significant number of parameters. These will be further studied in Chapter III and effectively eliminated in Chapter IV.

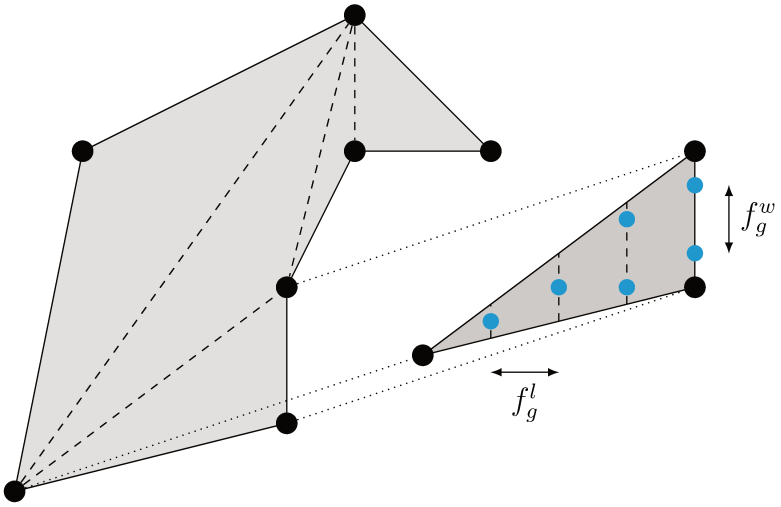


Figure II.3: Illustration for the ground polygon sampling procedure

Structural entities

Once again, structural entities are divided into three groups: First, structure points represent traffic lights and electric poles, among other things: they are nodes of interest on which cameras can be set up. Then come structure lines which correspond to walls and other similar elements. Finally, structure polygons describe buildings.

The process once again starts by identifying locations of interest, only this time, two more elements must be identified: a normal vector which will serve as the starting axis for pan sampling, and an angle which bounds the process. Structure nodes consist of a single location of interest, with a normal vector set to an arbitrary vector parallel to the ground, and a pan angle of 360 degrees. Structure lines are walked just like ground lines, although at each stop the algorithm registers two configurations: one of each side of the OSM way with the associated normal vectors and pan angles of 180 degrees. Note that this requires a new parameter, f_s^l, which represents the sampling frequency along a wall.

Structure polygons require a little bit more thought since a lot of them will share walls or corners. In order to avoid sampling along those and to determine valid pan angles at every location, we propose the following approach, illustrated in Figure II.4. The algorithm proceeds iteratively on triplets of consecutive nodes: previous, current and next until all nodes have been set as current once. For every triplet, two angles are first computed: α_p and α_n: they represent the available openings between the building's walls and those of any other structure. This is achieved by keeping track

of node-to-way mappings in the OSM data. Whenever one angle is zero, then the associated wall (current to previous or current to next) is shared and should not be sampled. When possible however, configurations are registered much like for structure lines, with only the outward normal vectors considered. The α_p and α_n angles also determine whether or not cameras can be placed at the current node itself, both towards the previous and next nodes. They then serve as pan angle values, and their bisector is used as the normal vector for sampling (again, parallel to the ground).

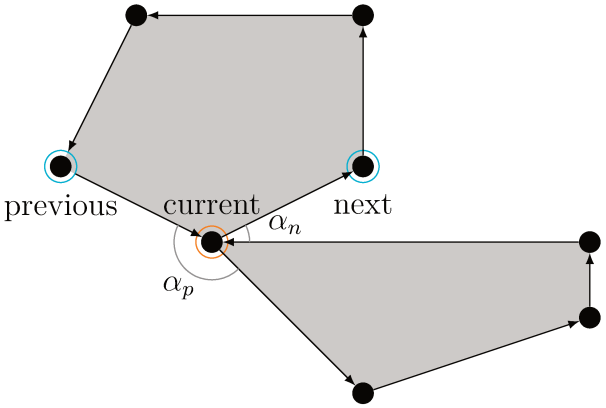


Figure II.4: Illustration for the structure polygon sampling procedure

At this point, our algorithm has only registered possible locations, with the associated normal vectors and pan angles. Just as we did in the previous section, we begin the actual sampling by duplicating those locations at several altitude values, this time using parameters a_g^{min}, a_g^{max} and f_s^a, similar to those used earlier for ground sampling. This time however, these parameters take into account the building's height configuration to ensure no camera is placed in the air.

Pan angle sampling may then begin. The available angle is divided into smaller sectors using parameter f_s^p (typically, a fraction of π) again centring the subdivision by padding away from the wall. Whenever the available angle is too small for the selected parameter, a single pan angle is chosen at the bisector. For every selected pan vector, it is then possible to compute the final orientation samples using parameter f_s^t for the tilt angles. Here, the available interval is given by the pan vector, parallel to the ground, and the wall, orthogonal to it. The available tilting room is therefore always $\frac{\pi}{2}$ and experiments suggest that the tilting frequency should not be set above $\frac{\pi}{6}$ to ensure optimisation algorithms are given enough options to chose from (see Chapter III).

Visibility analysis and online instance reduction

With both ground samples and camera candidates now available (see Figure II.5 for a visual example), we are now able to compute the visibility matrix which maps them together. In other words, for each sample i and each candidate j, we compute the binary value a_{ij} which is set to one if and only if j sees i in our surveillance area. As we have seen in Section I.2, the literature includes a wide variety of options for this process. One issue however is that those tend to be at both extremes of the complexity spectrum, using either simple two-dimensional geometrical tests or full-blown game engines or ray tracing algorithms.

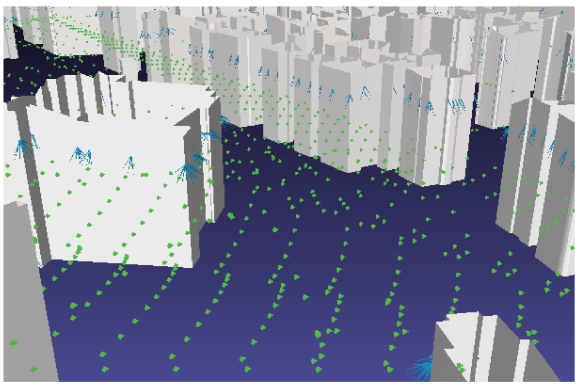


Figure II.5: Result of the sampling procedure for a subarea of the city of Strasbourg, France. Ground level does not match building base level in renders.

As will be seen later on, our instance generation process tends to create very heavy instances as it is able to tackle large surveillance areas. It would therefore be very impractical to use expensive visibility analysis algorithms which would require way too much time and scale poorly. Since realism is one of the main foci of our work however, we cannot simplify the process too much. As a middle ground, we propose the following approach. First, we define three consecutive tests to be applied to sample-candidate pairs, with one failure short-circuiting the sequence. The first is a simple range test: the distance between the sample and the candidate is measured, and the test succeeds if that value is below the range of the camera model. This range r is computed beforehand using Equation (II.8), which depends on the model's horizontal resolution R_h in pixels, its horizontal field of view f_h in radians and an operational parameter P which enforces an image quality constraint in pixels-per-meter. This approach was borrowed from earlier work on optimal indoor camera placement [24] although, as we will mention later, we have used much more tolerant

values for P. It is indeed not the goal of our research to design CCTV infrastructure for facial recognition or similar image processing pipelines. Our focus is on crowd analysis, for which high-resolution video capture is not really desirable (and in fact requires more constraints to be considered, many of them legal).

$$r = \frac{R_h}{2P \cdot tan\left(\frac{F_h}{2}\right)} \tag{II.8}$$

The second check to be performed relates to the orientation of the candidate. We have chosen to use oblique square pyramids to represent a candidate's frustum, with the orientation vector aligned with the segment which connects the apex to the centre of the base. When projected onto the ground plane, the frustum has a parallelepipedic shape and therefore takes into account the blind spot problem mentioned in Section I.2. Our use of this particular model was inspired from work by Fu, Zhou, and Deng [77] to which the reader is referred for more details. Our implementation however differs and tends to be more intuitive given our use of vectors for candidate representation. To check whether or not a sample lies within a candidate's frustum, we begin by computing the four side planes of the pyramid (see Figure II.6). To do so, we first compute the vertical plane in which the orientation vector lies. We use a dot-normal notation for planes and attach them to the candidate's mounting position. This first plane is then rotated by half the camera's horizontal field-of-view angle around the vertical axis at the candidate's position, once in each direction, to yield the left and right planes. For the other two planes, we start with the vertical plane from earlier and rotate it around the candidate's orientation vector by $\frac{\pi}{2}$ radians. We then compute the second axis of rotation by acquiring a vector orthogonal to the orientation vector in this plane. We then rotate around that vector by half the camera's vertical field-of-view angle in both directions to acquire the top and bottom planes. For these computations, we make sure to proceed such that each plane's normal vector is oriented inwards. The relative position of the ground sample is then easily checked with four dot products between each plane's normal and the vector which connects the attach points to the sample. If all four dot products are positive, the point lies within the camera's frustum. Note that we do not compute the pyramid's base plane at any point. This is unnecessary thanks to the previous range check. It is also very convenient since the pyramid's base can have almost any shape now that the ground has been elevated.

If we reach this point, we know that in an empty environment, our candidate would cover our sample. This brings us to our last check: occlusion. With trees, walls, street corners and other elements taken into account, there is no certainty that the line of sight between the candidate and the sample is clear. As we have seen

(a) Top view (outside, towards the top plane)

(b) Side view (outside, towards the left plane)

Figure II.6: Computing bounding planes for a vector-defined square pyramid frustum

earlier, several authors have chosen to use complex ray-tracing algorithms to tackle occlusion. To save on computational cost, we have decided to reuse the data we already have and generate a three-dimensional triangular mesh of all registered sources of occlusion in the surveillance area, which can easily be identified and reconstructed using a set of filter rules on OSM tags (see Figure II.7 for an example). This idea for using meshes had already been applied to the AGP by Fantini and Chaimowicz [68] who mentioned an implementation for the Half-Life game and the Illricht game engine [99]. For our implementation, we decided to use the available OSM data and the Möller-Trumbore algorithm for ray-triangle intersection [139]. Our first experiments used OSM2World [156] for mesh generation, which had been extended to use our coordinate system, however we integrated the process into our code base later on.

For each pair which passes the range and frustum checks, we go through the available mesh faces and look for intersections with the candidate-sample segment. When no such intersection can be found, the occlusion check passes and the pair is registered. As the reader will probably notice, this process can still be quite expensive if implemented in a straightforward manner. Iterating over all ground samples and all camera candidates would indeed make the algorithm extremely sensitive to the size and complexity of the surveillance area. We can however significantly alleviate this computational burden with two tricks: localised visibility analysis and online instance reduction. The former starts with a simple observation. As the surveillance area grows, the number of possible samples for a candidate increases, however the proportion of positively checked pairs decreases. For every new ground sample, only the pairs which involve nearby candidates will be validated, while a huge number of others will fail the range check. The same can be said about occlusion faces, with only a small subset being of interest for every pair. As it so happens, even this simple

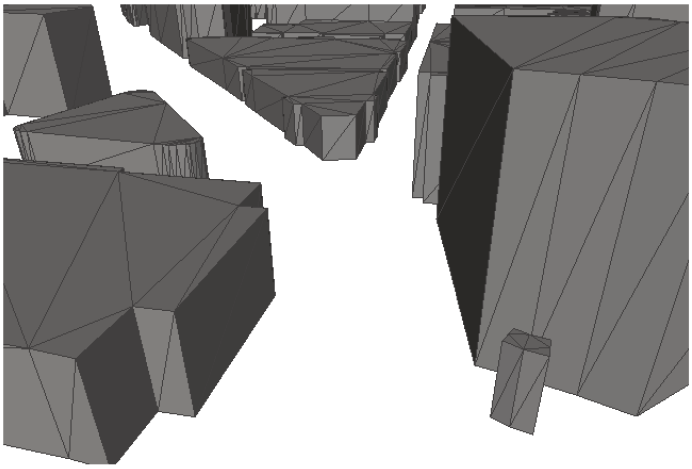


Figure II.7: An example of an occlusion mesh built using OSM data

check can turn out to be very costly as instances scale up, not to mention the even simpler instructions related to iterating over such large sets. We have been able to eliminate this overhead by running an entity-oriented analysis. Rather than iterate over candidates after the sampling procedure has completed, we instead iterate over the OSM elements (buildings, walls, ...) which yield them. For every structure entity, we begin by computing its bounding box and extending it in all directions by the camera model's range r. This box is then used as a query to an r-tree spatial index data structure which contains all ground entities (a similar query is performed to fetch the relevant occlusion faces). This allows us to quickly acquire a list of all nearby ground entities which may have samples within the aforementioned coverage box. We can then sample both the structure entity and the local ground entities to yield camera candidates and ground samples. The analysis is then performed on those small subsets only, since any other pair for the current structure entity would necessarily fail the range check. After comparing with straightforward implementations, the advantage of r-trees has proven to be major, beating even our initial massively parallel implementation on GPU devices. The reader is referred to [89] by Guttman for more information about these spatial indices and their performance. For our work, we have chosen to use the Boost.Geometry implementation [21] with the R^* balancing algorithm [16].

As the visibility analysis validates pairs, those need to be stored for further processing, including instance reduction. Given the algorithms which we plan to apply to this problem, the following characteristics are desirable. First, iterating over a candidate's sample set should be quick. Similarly, we should be able to efficiently

access all covering candidates for a sample. Samples and candidates should therefore ideally be consecutively indexed for a random-access container, which should contain ordered coverage sets on which intersections and differences can be efficiently computed. The first approach which comes to mind is the use of maps (or any other indexed tree-like structure). One would connect candidates to their samples and another would map entries in the other direction. This however has two drawbacks: insertion is very costly and reduction would require even more costly reindexing operations if we are to have constant-time access to our coverage sets. Additionally, this structure would not exactly agree with the reduction algorithms, which require sets to be sorted by size rather than by candidate index.

We instead propose the following approach: when a pair is validated by the analysis, we store it as a tuple of indices in a random-access container. After a first candidate sort, we can tag each pair with the size of its candidate's cover. Domination and inclusion checks can then be performed rather efficiently by repetitively applying a quicksort algorithm [92]. The process for both reduction checks is described in Algorithms 1 and 2. It should however be noted that these two procedures affect each other. For example, a reduction performed while checking for inclusion can create the conditions necessary for additional domination reduction. For this reason, these procedures are to be repeated until there are no more changes.

Once all pairs have been processed, samples and candidates can be efficiently reindexed by sorting the container twice again. It can then be transformed into actual sets, which we store in random-access containers again. The final data structures allow us to access the cover set for any sample or candidate in constant-time and iterate over said sets in linear time. The analysis process itself will have used constant-time insertions while the final reindexing and packing into sets can be brought down to linear time thanks to the repetitive use of the quicksort algorithm. The latter of course performs in logarithmic time on average and linear time in worst-case scenarios. The overall process as it was first implemented is summarised in Algorithm 3.

As we started working on even larger surveillance areas however, we noticed that the space complexity of the first phase (pair registration) became unmanageable. We started storing this data using disk-backed, memory-cached containers but the I/O overhead also became impractical eventually. This is where our second trick comes in: online instance reduction. The important observation here has to do with the way the visibility analysis loops are structured: pairs will not be registered in a random order, but rather one structure entity at a time. This means that consecutive pairs are very likely to involve nearby candidates and samples. In other words: pairs are being processed in a sensible positional order, one area of the map at a time. It can also be noted that once we move on to another candidate, we are certain no

more pair will appear for the previous one. This is a very useful piece of information for the domination checks: pairs are coming in such that two consecutive candidate blocks actually correspond to nearby areas and therefore have a high probability of dominating one another. This enables us to save a lot of time on instance reduction by performing partial domination checks as the analysis is being performed, and a final domination check on the pairs at the end. The overall process, to be run with C CPU cores available, is best described using pseudocode, which we provide in Algorithm 4 and illustrate in Figure II.8.

II.3 First instances and observations

For our first set of tests, we selected 8 areas extracted from the following 8 European cities, most of them home to fellow OPMoPS collaborators: Berlin, Kaiserlautern, London, Munich, Rennes, Mulhouse, Valbonne and Mainz. The area codes as well as the associated OpenStreetMap identifiers are summarised in Table II.1. The symbol column refers to a code we will be using later on to label our instances.

OSM ID	Symbol	Location (postal code)
1402158	b	Berlin Mitte (10117)
1186690	k	Kaiserslautern (67655)
51800	l	City of London (various)
1100773	m	München city centre (80335)
6796357	r	Rennes city centre (various)
Custom	u	Mulhouse city centre (various)
92482	v	Valbonne (06560)
1236525	z	Mainz city centre (55118)

Table II.1: The 8 locations used as our first test set

For these instances, our filtering rules for sampling and occlusion handling were rather simple. Anything tagged as a road or an open public area such as a park or pedestrian way was registered as a ground entity. Buildings, walls and poles were used as structure ways, and the first two also served as the basis for the construction of the occlusion mesh.

As we mentioned earlier in Chapter II, generating an instance from an area also requires us to set sampling parameters. In order to understand their influence better, we created a set of 4 sampling configurations, which we indexed from 0 (superficial) to 3 (intensive). The corresponding parameter values are reported in Ta-

bles II.2 and II.3. The camera model used has resolution 1920×1080 and is expected to perform at 25 pixels-per-meter.

Cfg.	f_g^l	f_g^w	f_g^a	a_g^{min}	a_g^{max}
0	10	2	1	0	1
1	7	2	1	0	1
2	5	1	1	0	2
3	3	1	1	0	2

Table II.2: First set of ground sampling parameter values

Cfg.	f_s^l	f_s^a	a_s^{min}	a_s^{max}	f_s^p	f_s^t
0	6	1	3	3	$\frac{\pi}{3}$	$\frac{\pi}{4}$
1	5	1	3	3	$\frac{\pi}{4}$	$\frac{\pi}{4}$
2	4	1	3	4	$\frac{\pi}{5}$	$\frac{\pi}{6}$
3	3	1	3	4	$\frac{\pi}{6}$	$\frac{\pi}{6}$

Table II.3: First set of candidate sampling parameter values

We labelled our instances using the symbol for the area suffixed with the configuration number such that, for example, applying configuration 3 to the Berlin area yields instance `b3`. The generation was performed at the Tier 2 Strasbourg high-performance computing cluster [96] using 2.5-2.67GHz Intel processors. The basic statistics for this first set of 32 instances are reported in Table II.4. The last three columns provide the numbers of ground samples and camera candidates, as well as the density of the visibility matrix. All instances have been reduced.

The first observation to be made is that of scale. To date, the only widely recognised benchmark for set covering problems is the OR Library [13] which we mentioned in Chapter I. Looking at the statistics, our instances are clearly much larger. In fact, only the 4 largest RAIL instances exceed our own statistics, and only so in regards to set $\mathcal{J}$. It should also be noted that these instances have not been reduced. When it comes to density however, our values are several orders of magnitude below that of the benchmark.

In order to get a first idea of the solution landscape for our instances, we ran three basic solving algorithms. The first is random: it processes ground samples in a random sequence and for each of them selects a random candidate, unless the sample is already covered. The second is greedy and selects the most interesting

candidate (ie. the one which brings the most new samples into the solution) at every iteration, breaking ties randomly. The last one is CPLEX's default configuration, which involves a Branch-and-Cut algorithm at its core, which we stopped after 1 hour (we will label this as BIP for Binary Integer Program). The random and greedy algorithms were both run 30 times and the average solution size was computed. Table II.5 reports on the results. The last two columns are the difference between the BIP and greedy results, which we take as a measure of the effective solving range. Indeed, the greedy algorithm is among the most time-efficient approaches, while the BIP strives for optimality, which makes it fitness-efficient. In the last column, we scale this absolute gap over the size of $\mathcal{J}$.

As we can see, the improvement on the solutions is relatively small when we move from a greedy to a one hour BIP solve, especially given that CPLEX reported duality gaps below 10% (many below 5%) for all instances except `b3`. Since the greedy algorithm performs much more time-efficiently, it therefore appears as though striving for optimality on our instances comes at very high computational cost and does not yield much of a reward. The results also suggest that our instances are much simpler to solve than standard benchmarks or previous work using high-resolution, uniform sampling grids by Brévilliers et al. [24]. In the latter, many of the instances remained unsolved by CPLEX. The only clear element of difference between this work and ours is the average instance density. It would appear that our instance generation framework yields very low-density matrices which severely lower the curse of dimensionality for the problem. In Chapter III, we take a closer look at this observation and bring forward hypotheses which will help us approach the problem from a different angle.

Review

Being able to create instances from open real-world data presents two main advantages. First, our process has no dependency whatsoever on limited-access data sources and is designed to work on an area without any data preprocessing required. Second, we ensure that the instances are always tailored for their study cases and avoid making assumptions regarding the area's geometry or complexity. In this chapter, we therefore introduced such an instance generation framework and covered coordinate system changes, ground sampling, camera candidate sampling, occlusion modelling, as well as efficient visibility analysis and reduction algorithms.

Instances were then generated using several cities involved in our research project. From these, we were able to acquire a first picture of the solution landscape when

using a full-coverage set covering model, as has so far typically been done in the literature. After running several basic algorithms, we noticed a significant difference in complexity between our instances and previous works. We suspect this is due to our generation framework and our particular application, which yield simpler, lower-density instances. We believe interesting hypotheses could stem from this knowledge and may reveal new solving approaches, perhaps better suited to the specifics of our research project. We explore these possibilities in Chapter III.

Symbol	Cfg.	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{J}\rvert$	Density
b	0	9645	3854	0.0018
	1	12526	4301	0.0018
	2	168773	51245	0.0010
	3	288968	87287	0.0010
k	0	8610	3829	0.0014
	1	12374	4548	0.0013
	2	158528	53003	0.0010
	3	263767	87843	0.0009
l	0	13881	5800	0.0012
	1	19940	7608	0.0011
	2	190677	61722	0.0008
	3	309447	98309	0.0008
m	0	6862	2846	0.0028
	1	7978	2861	0.0034
	2	112457	33517	0.0017
	3	193275	54800	0.0016
n	0	2678	1160	0.0059
	1	2778	967	0.0087
	2	42209	12458	0.0044
	3	69902	19625	0.0040
u	0	900	446	0.0131
	1	1688	675	0.0100
	2	15926	6255	0.0073
	3	28927	10898	0.0072
v	0	7840	3132	0.0019
	1	11337	4054	0.0017
	2	214874	47462	0.0006
	3	374447	79237	0.0005
z	0	5027	1618	0.0042
	1	6218	1752	0.0047
	2	95283	20952	0.0022
	3	155398	32992	0.0022

Table II.4: Basic statistics for our first set of 32 real-world instances

Symbol	Cfg.	Random	Greedy	BIP	Greedy-BIP gap	
					Absolute	Relative to $\|\mathcal{J}\|$
b	0	1176.53	845.03	774	71.03	0.0184
	1	1227.07	875.03	796	79.03	0.0184
	2	2685.73	1597.40	1285	312.40	0.0061
	3	2971.30	1703.20	1412	291.20	0.0033
k	0	1350.10	982.03	922	60.03	0.0157
	1	1516.50	1095.97	1014	81.97	0.0180
	2	2797.80	1630.17	1311	319.17	0.0060
	3	2941.87	1661.93	1354	307.93	0.0035
l	0	1832.13	1305.17	1202	103.17	0.0178
	1	2153.03	1519.27	1397	122.27	0.0161
	2	3909.20	2287.27	1901	386.27	0.0063
	3	4285.10	2446.17	2002	444.17	0.0045
m	0	805.80	563.33	516	47.33	0.0166
	1	747.97	526.37	484	42.37	0.0148
	2	1799.77	1038.40	851	187.40	0.0056
	3	2046.63	1137.77	958	179.77	0.0033
n	0	371.30	268.93	249	19.93	0.0172
	1	281.13	203.27	188	15.27	0.0158
	2	723.50	429.23	340	89.23	0.0072
	3	776.53	448.23	361	87.23	0.0044
u	0	147.17	106.47	101	5.47	0.0123
	1	199.43	136.50	130	6.50	0.0096
	2	367.93	208.27	167	41.27	0.0066
	3	406.73	226.53	177	49.53	0.0045
v	0	1033.73	747.53	693	54.53	0.0174
	1	1177.70	829.43	760	69.43	0.0171
	2	4101.43	2490.60	2061	429.60	0.0091
	3	4673.73	2751.53	2240	511.53	0.0065
z	0	528.73	375.03	349	26.03	0.0161
	1	507.20	362.23	338	24.23	0.0138
	2	1243.70	749.27	596	153.27	0.0073
	3	1309.77	762.07	609	153.07	0.0046

Table II.5: First basic solving results for the first set of real-world instances

Algorithm 1 Our implementation of domination checks on sample-candidate pairs

```
1: procedure DOMINATION(P, n)                          ▷ Vector of pairs and its size
2:     sort P by ascending candidate cover size, ascending candidate index
3:     β_start ← 0
4:     while β_start < n do
5:         β_end ← β_start
6:         while candidate(P[β_start]) = candidate(P[β_end]) do
7:             β_end ← β_end + 1
8:         end while
9:         if any pair has been marked unnecessary between β_start and β_end then
10:            continue to the next iteration
11:        end if
12:        removed ← false
13:        α_start ← β_end
14:        S_β ← all sample indices which appear between β_start and β_end
15:        while removed = false and α_start < n do
16:            α_end ← α_start
17:            while candidate(P[α_start]) = candidate(P[α_end]) do
18:                α_end ← α_end + 1
19:            end while
20:            if any pair has been marked unnecessary between α_start and α_end then
21:                continue to the next iteration
22:            end if
23:            S_α ← all sample indices which appear between α_start and α_end
24:            if S_β ⊆ S_α then
25:                mark all pairs between β_start and β_end as unnecessary
26:                removed ← true
27:            end if
28:        end while
29:    end while
30: end procedure
```

Algorithm 2 Our implementation of inclusion checks on sample-candidate pairs

```
1:  procedure INCLUSION(P, n)                          ▷ Vector of pairs and its size
2:      sort P by sample index
3:      i ← 0
4:      J_inc ← ∅                                      ▷ Building the set of included candidates
5:      while i < n do
6:          i' ← i
7:          c ← 0
8:          j ← candidate(P[i'])
9:          while sample(P[i]) = sample(P[i']) and i < n do
10:             if pair i is not marked as unnecessary then
11:                 j ← candidate(P[i])
12:                 c ← c + 1
13:             end if
14:             i ← i + 1
15:         end while
16:         if c = 1 then
17:             J_inc ← J_inc ∪ {j}
18:         end if
19:     end while
20:     I_aff ← ∅                                      ▷ Recording all directly affected samples
21:     P' ← {p | p < n, candidate(P[p]) ∈ J_inc}
22:     for p ∈ P' do                                  ▷ Marking directly affected pairs
23:         mark P[p] as unnecessary
24:         I_aff ← I_aff ∪ {sample(P[p])}
25:     end for
26:     P'' ← {p | p < n, sample(P[p]) ∈ I_aff}
27:     for p ∈ P'' do                                 ▷ Marking indirectly affected pairs
28:         mark P[p] as unnecessary
29:     end for
30: end procedure
```

Algorithm 3 A simple sequential visibility analysis implementation

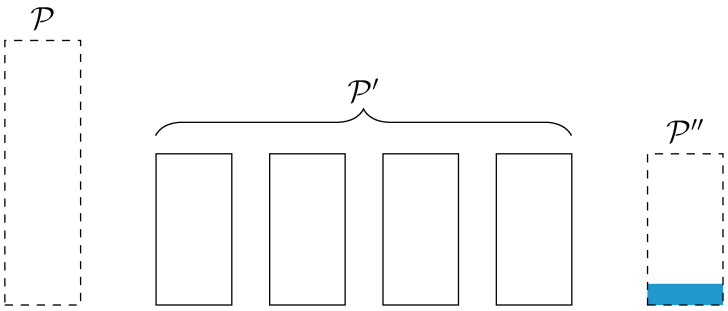
```
1: procedure SequentialVA(G, S, F)          ▷ Ground and structure entities, occlusion faces
2:     P ← ∅
3:     n ← 0
4:     for s ∈ S do
5:         b ← the bounding box for s, extended by r
6:         G′ ← all ground entities which intersect b
7:         F′ ← all occlusion faces which intersect b
8:         J ← the camera candidates yielded by s
9:         I ← the ground samples yielded by the entities in G′
10:        for j ∈ J do
11:            for i ∈ I do
12:                if all 3 visibility checks pass for i and j given F′ then
13:                    P ← P ∪ {(i, j)}
14:                    n ← n + 1
15:                end if
16:            end for
17:        end for
18:    end for
19:    while the reduction checks alter P do
20:        Domination(P, n)
21:        Inclusion(P, n)
22:    end while
23:    reindex P and generate cover sets
24: end procedure
```

Algorithm 4 A more efficient online visibility analysis implementation

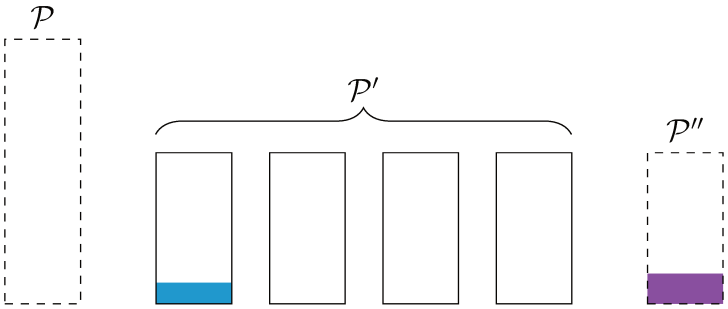
$P \leftarrow \emptyset$ ▷ Global pair container
$P' \leftarrow \{\emptyset\}^C$ ▷ Secondary pair containers (limited capacities)
$P'' \leftarrow \emptyset$ ▷ Temporary candidate buffer

procedure OnlineVA(G, S, F) ▷ Ground and structure entities, occlusion faces
  for $s \in S$ **do**
    $b \leftarrow$ the bounding box for s, extended by r
    $G' \leftarrow$ all ground entities which intersect b
    $F' \leftarrow$ all occlusion faces which intersect b
    $J \leftarrow$ the camera candidates yielded by s
    $I \leftarrow$ the ground samples yielded by the entities in G'
    for $j \in J$ **do**
      for $i \in I$ **do**
        if all 3 visibility checks pass for i and j given F' **then**
          RegisterPair(i, j)
        end if
      end for
    end for
  end for
  for $P'_{full} \in P'$ in parallel **do**
    Domination(P'_{full}, $|P'_{full}|$)
    $P \leftarrow P \cup P'_{full}$
    $P'_{full} \leftarrow \emptyset$
  end for
  $P \leftarrow P \cup P''$
  while the reduction checks alter P **do**
    Domination(P, $|P|$)
    Inclusion(P, $|P|$)
  end while
  reindex P and generate cover sets
end procedure

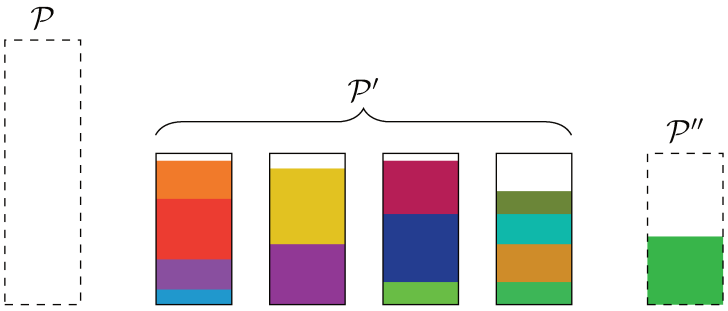
procedure RegisterPair(i, j)
  if P'' is not empty and the pairs in P'' do not involve j **then**
    $P'_{open} \leftarrow$ the first buffer in P' with sufficient capacity to store P''
    if no viable P'_{open} could be found **then**
      for $P'_{full} \in P'$ in parallel **do**
        Domination(P'_{full}, $|P'_{full}|$)
        $P \leftarrow P \cup P'_{full}$
        $P'_{full} \leftarrow \emptyset$
      end for
      $P'_{open} \leftarrow P'[0]$
    end if
    $P'_{open} \leftarrow P'_{open} \cup P''$
    $P'' \leftarrow \emptyset$
  end if
  $P'' \leftarrow P'' \cup \{(i, j)\}$
end procedure



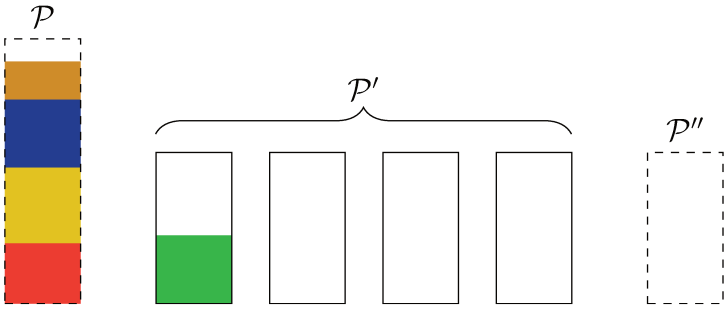
(a) Step 1: `REGISTER_PAIR` enqueues pairs for a candidate (blue) into $\mathcal{P}''$



(b) Step 2: pairs for a new candidate (purple) arrive in $\mathcal{P}''$, the previous contents are moved into the first $\mathcal{P}'$ vector which can hold them



(c) Step 3: the process continues until all $\mathcal{P}'$ vectors are full



(d) Step 4: domination checks are performed in parallel in each $\mathcal{P}'$ vector, the pairs for the dominating candidates are moved to the main vector $\mathcal{P}$ and the process repeats

Figure II.8: An illustration for Algorithm 4

Chapter III

Stress-testing and validation

Introduction

Now that the problem has been given a basic formulation and preprocessing stages, we can begin to focus on the various aspects of solving. In the previous chapter, we concluded by providing statistics for our instance set and running three basic algorithms to get a first picture of the solution landscape. It was suggested that solving the problem based on a standard set covering model might not be the most efficient approach and that our instances appear to display application-specific characteristics which make it less sensitive to the curse of dimensionality. In this chapter, we first go further along those lines and run a uniform benchmark of the state of the art, as reviewed in Chapter I and applied to optimal camera placement. Due to the wide variety of variants around the problem, and to the best of our knowledge, no such analysis has ever been performed. Based on our results, we categorise existing approaches and attempt to identify efficient algorithmic components to be reused. We also formalise two hypotheses about our instances (and more generally our application) which we validate through independent simulations in continuous space. From this study, we begin to outline a human-assisted optimisation approach which we will develop further in Chapter IV.

Related publications

[120] Julien Kritter et al. "On the real-world applicability of state-of-the-art algorithms for the optimal camera placement problem". In: *6th 2019 International Con-*

ference on Control, Decision and Information Technologies (CoDIT). IEEE, Apr. 2019. DOI: 10.1109/CoDIT.2019.8820295
[118] Julien Kritter et al. "On the computational cost of the set cover approach for the optimal camera placement problem and possible trade-offs for surveillance infrastructure design". In: *RAIRO Operations Research* (2020). Submitted

III.1 A stress-test for the state of the art: hypotheses

In the previous chapter, we used three basic algorithms in order to evaluate the ranges between typical good and bad solutions for our instances. These tests revealed that there is in fact very little room for fitness improvement between a greedy and an exact solve.

In this first section, we begin by extending those results by performing a complete benchmark of OCP and unicost SCP state-of-the-art literature on our instances. As we have seen in Chapter I, such an analysis has never really been performed in the past: with OCP variants being so numerous, comparing them fairly without establishing a new test-bed is bound to result in some bias at some point. Because we have formulated our problem as a set cover application from the start however, our instances represent a good opportunity for such a test. Additionally, the results will allow us to compare more tailored approaches and hopefully identify algorithmic components which perform well on the OCP and could be further exploited, a study which is much harder to perform using only greedy or branch-and-bound algorithms.

Based on the study presented in Chapter I, we propose to evaluate the following algorithms. First, we extend the baseline acquired in Chapter II by replacing the greedy algorithm with its non-deterministic variant GRASP [72], which can be tuned through its $\alpha \in [0; 1]$ parameter (from absolute randomness to absolute greediness). We ran the algorithm for $\alpha \in \{1, 0.7, 0.5\}$. For an actual lower (dual) bound, we are also including a linear relaxation solve performed by CPLEX.

We then included five algorithms from OCP and unicost SCP literature, which we selected based on their originality and reported performance (our objective being to study various algorithmic components).

- Marchiori and Steenbeek's ITEG algorithm as referenced by Andersen and Tirthapura [4] for their work on 3D sensor placement ;
- For its efficiency on the OR-Library CLR instances, Meta-RaPS by Lan, DePuy, and Whitehouse [123] ;

- For the STS instances, 3-FNLS by Yagiura, Kishida, and Ibaraki [191], the code for which was generously provided by the authors ;
- As the overall best to date, RWLS by Gao et al. [79] ;
- Finally, the OCP DEsim algorithm by Brévilliers et al. [25] for its efficiency on toy OCP instances for global area surveillance.

Every algorithm was given one hour on Intel Xeon processors to yield their best solutions. Nondeterministic methods were ran 30 times each for better statistical reliability. Parameter setting was left to the respective authors' discretion based on the results they reported. Figure III.1 reports on the average results obtained by every algorithm on our instance set.

Starting with the baseline and without too much surprise, the random algorithm provides a rather generous upper bound on every instance and confirms the appeal in designing optimisation algorithms for this problem. It also shows a significant standard deviation between runs, which makes sense for random guessing.

Regarding GRASP, the algorithm shows poor performance when compared to the rest of the benchmark, but nevertheless provides a much tighter upper bound on the problem, which might explain why so many propositions use a greedy for initialisation. Slightly more surprising however are the variations between the different GRASP parameter values. Indeed, while GRASP was originally introduced as an improvement over Chvátal's greedy algorithm [40], such a conclusion cannot be drawn using our instances: the three sets of results are extremely similar. Looking at both best and average results, the pure greedy variant slightly outperforms the others on 13 (best) and 12 (average) instances. The $\alpha = 0.5$ and $\alpha = 0.7$ setups rank next with 11 instances on both measures for the former and between 8 (best) and 9 (average) for the latter.

Looking at Figure III.1, a first observation is that there is very little variation between algorithms in configurations 0 and 1. All algorithms are indeed able to settle between GRASP's upper bound and the linear relaxation. Regarding average results in configurations 2 and 3 however, the divide in performance is clearer and highlights the performance of CPLEX, RWLS and DEsim. More precisely, CPLEX and RWLS both report the best average on 25 out of 32 instances and cover all the instances when taken together, while other algorithms could only do it for instances `u0` and `u1`. DEsim managed to stay reasonably close behind and joined the rankings for one more instance. Intermediary instances such as `u2`, `u3` and `n2` also isolate ITEG and MetaRaPS in another cluster, however these algorithms still report poorer performance at a larger scale.

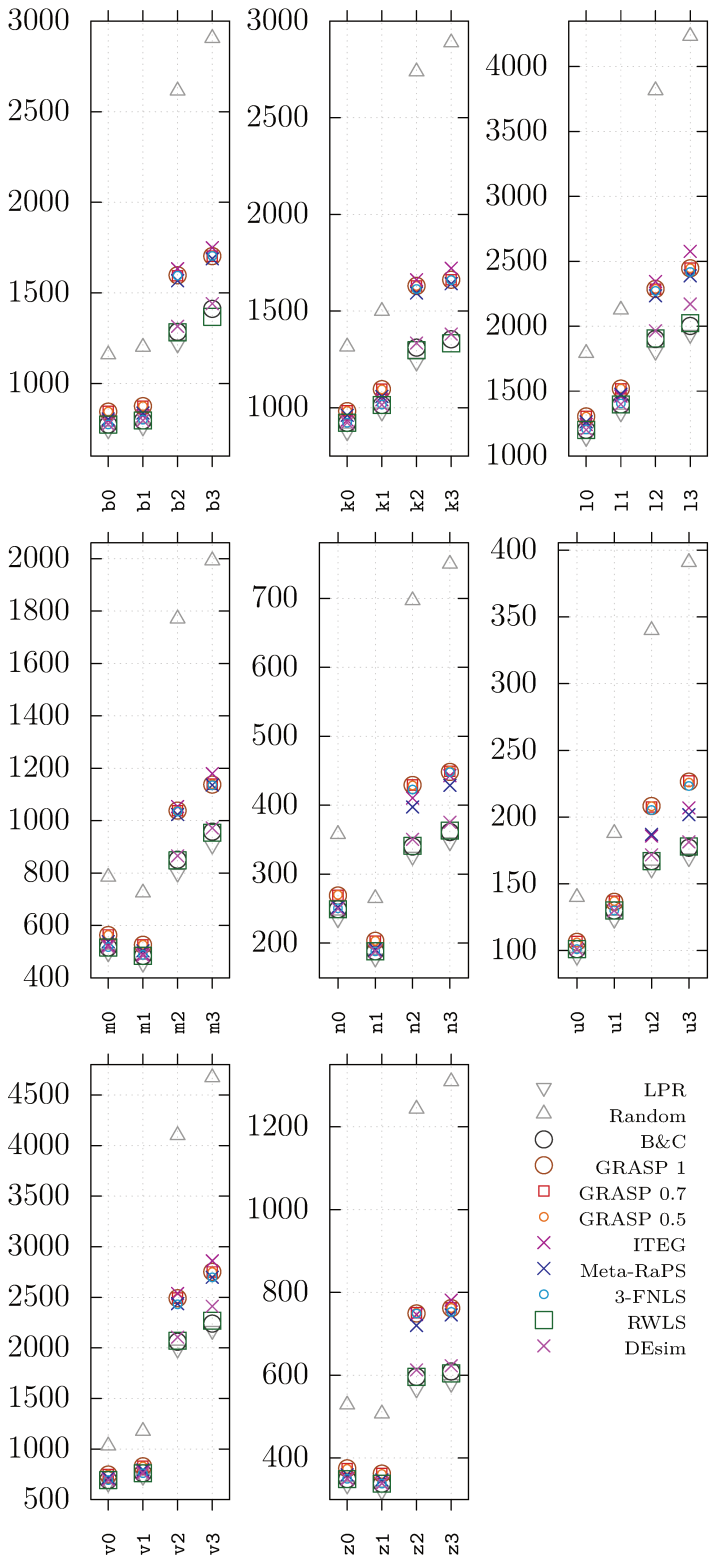


Figure III.1: Average state-of-the-art results for every instance from our first set

Some of these results may seem surprising for a problem we might have assumed to be more severely cursed by $\mathcal{NP}$-hardness. Experiments conducted on toy instances so far had actually shown the problem to be extremely sensitive to dimensionality, and to quickly exhaust a Branch-and-Cut algorithm, sometimes to a point where no feasible solutions could be found [95, 63, 25]. For 25 out of 32 instances however, CPLEX ranked first and reached global optimum on 16. More importantly, it managed to find feasible solutions to all instances, something it could not always achieve on the OR-Library instances [13] or the toy instances used in [25, 24]. Most of these sets are significantly smaller than the instances used in this paper when looking at $|\mathcal{I}|$ and $|\mathcal{J}|$, however a key difference which may explain the results is our instance density, which ranges several orders of magnitude below of these benchmarks. Echoing the hypothesis suggested in Chapter II, this can be easily understood as follows: for a given candidate, the proportion of samples it can cover (w.r.t $|\mathcal{I}|$) is bound to be very low and get lower as the surveillance area grows. In other words, for every sample, the set of eligible candidates does not depend so much on the size of the surveillance area as it does on the sampling frequencies used in the instance generation process. This means that for every constraint of the BIP formulation, a solver only has a very limited set of variables to work with.

These algorithms bring forward the following hypotheses. First, candidate weighting schemes alone seem insufficient and can bring algorithms closer to greedy results as the instances grow in size. ITEG, Meta-RaPS and 3-FNLS all use such approaches. The results of the latter also suggest that the usually-efficient Lagrangian framework is less appealing for unicost low-density instances, even though the algorithm was able to obtain reasonable best results. In any case, RWLS proved efficient and also uses a weighting scheme, but this time focuses on rows rather than columns. The algorithm seems to identify samples which are easily lost and favours them as it rebuilds its solution. DEsim on the other hand seems to make good use of its similarity measure, suggesting that a good solution may often be neighbouring several others. This hypothesis appears to be supported by CPLEX, which reports low duality gaps and suggests that good and optimal solutions may often be surrounded by many alternatives with similar costs, making the gap more difficult to close.

From these last few observations, the following elements appear to be of interest. First, the complexity of our instances appears to depend more on the sampling frequencies than it does on the size of the surveillance area. This is observed from the clear divide between configuration 1 and configuration 2, where the instance sizes jump and algorithm performance begins to deteriorate. So far however, we have no evidence that finer sampling frequencies are worth the additional computational cost. Since our model always reports 100% coverage, it is very difficult to truly determine

whether choosing configuration 3 over configuration 0 does indeed lead to better network performance in practice.

Second, it is suggested that a lot of the allocated computational budget is spent on minor improvements to the solutions (closing the gap). Algorithms appear to improve their incumbent solutions most significantly in the very early stages of their runs, and then spend the rest of the available time polishing. We believe this behaviour is strongly influenced by the full-coverage requirement. In other words, it is possible that we could significantly reduce our solving time and overall complexity by allowing for small relaxations of this constraint. Again, its impact has yet to be confirmed outside of the scope of our model.

Finally, two algorithmic components seem to be efficient when combined. The first are dynamic weighting schemes applied to either samples or candidates, which guide the algorithm's focus as it satisfies the coverage constraints. The second is that of keeping the algorithm very close to the feasibility edge by repetitively breaking and repairing solutions by small amounts. This coincides with previous observations about algorithms spending most of their budget on small improvements.

III.2 Measuring the impact of sampling frequencies

The first element we attempted to verify was the impact of our sampling frequencies, which appear to significantly impact algorithm performance and computational cost. In most cases, when sampling for continuous optimisation problems, the typical observation is that as the sampling resolution increases, the solutions improve and get closer to a limit which we could visualise as being the optimal solution to the original continuous problem. In this first section however, we show that this phenomenon does not really occur for optimal camera placement for area coverage when using a set cover model. While the solutions do use more candidates when the sampling is finer, this does not necessarily reflect an increase in quality when the solution is evaluated with regards to the application at hand. In other words: the additional cameras do not actually offer significantly more area coverage.

To visualise the actual impact of the sampling frequencies, we propose to use an agent-based simulation algorithm. For each city, we create a simulation scene which includes all known obstacles extracted from the available map data. This is similar (but simpler) to the process we used in Chapter II to generate occlusion meshes. Agents are then created and given a start location as well as a destination, both located close to building walls. A camera placement solution is also chosen and

the corresponding infrastructure is set up on the various structure entities available in the area. At every step of the simulation, the agents are allowed to move 1.4 meters, which is the average distance walked by a human in a second. Every step therefore represents one second in the simulated world. Once all agents have moved in an iteration, the visibility checks described in Chapter II are used to determine which agents are being recorded by at least one camera in the selected solution. Their number is then recorded as a percentage of the total number of agents and the simulation moves on to its next iteration. For all tests, we ran this algorithm for 3600 iterations, which correspond to one hour in the simulated world.

In order for the agents to find their way around the city, a routing system had to be designed. Our implementation is as follows. First, the geometry of all recorded obstacles is simplified: entities which share edges are ground together using a graph search algorithm. In graph theory, this correspond to the problem of identifying connected components. At every node in this graph, we then run a simple visibility check to determine which other nodes (across all components) can be reached. These pairs are registered as edges which we will call visibility edges, in contrast with wall edges which represent the connected components previously identified. The final result is a routing graph for our agents to follow. An simple example of such a graph is given in Figure III.2.

Because this graph is typically very dense, even more so as the OSM data gets more precise in large cities, it allows our agents to walk virtually anywhere in the area, even under the constraint that they should always follow the shortest path from their start locations to their destinations. This is achieved by using the A* path-finding algorithm [90, 54] in the routing graph for every agent, which will follow its optimal route until it reaches its destination and randomly respawns at another location for another trip. We determined the number of agents using a simple auto-scaling algorithm which computes it from the approximate outdoor area of the city and a given density parameter. The former is estimated by computing an alpha-shape of the area's nodes, a generalisation of the concept of convex hulls [60]. The indoor area, which is easily computed from the recorded buildings, is then subtracted. For our experiments, we used a density value of 0.0035 agents per square meter, a reasonable albeit empirical assumption which most importantly maintains uniformity across our instances.

Before running these simulations, we also adjusted our candidate sampling parameters. As we observed earlier, configurations 0 and 1 appear to be particularly simple to solve. We also observed that using a tilt frequency below $\frac{\pi}{6}$ led to many ground samples being impossible to cover, since the available candidates focused on areas near buildings and left larger open areas uncovered. The new set of candidate

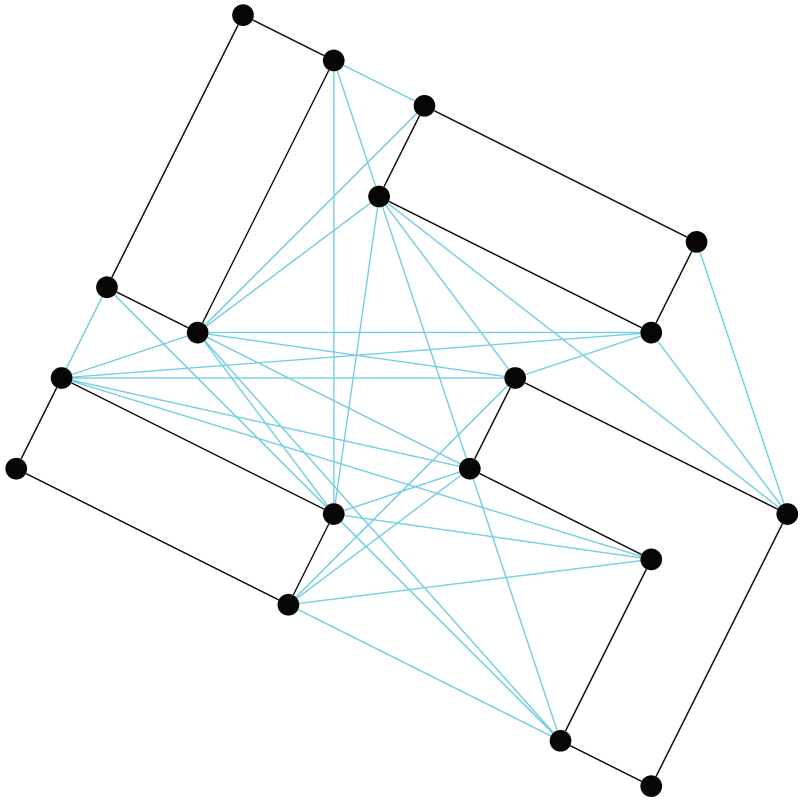


Figure III.2: An example routing graph with wall and visibility edges

sampling parameters is given in Table III.1. The ground sampling parameters remain unchanged (see Table II.2) although we will be using configuration numbers 4 to 7 from now on. Table III.4 reports on the basic statistics for our new instances. The reader will note that the Valbonne instances have been removed. This particular area contains a large forest which leads to the surveillance area being very empty: a lot of ground samples are impossible to reach because no structure entity can be found nearby. It should also be noted that for this part of our work, we removed the elevation data from our instance generation process to simplify the representation used in our simulation algorithm.

To study sampling frequencies, we conducted the following simulation experiment. For every instance, the greedy full-coverage set cover solution was computed and used to define the infrastructure set up in the simulated world. In each city, we then ran a simulation for each sampling configuration. The proportion of visible agents (effective coverage rate) was recorded at every step. Figure III.3 reports on

Cfg.	f_s^l	f_s^a	a_s^{min}	a_s^{max}	f_s^p	f_s^t
4	6	1	3	3	$\frac{\pi}{3}$	$\frac{\pi}{6}$
5	5	1	3	3	$\frac{\pi}{4}$	$\frac{\pi}{8}$
6	4	1	3	4	$\frac{\pi}{5}$	$\frac{\pi}{10}$
7	3	1	3	4	$\frac{\pi}{6}$	$\frac{\pi}{12}$

Table III.1: Second set of candidate sampling parameter values

our results.

As we can see, the effective coverage rate barely increases with the sampling frequencies: the difference between configurations 4 and 7 is barely perceptible for all our instances. The instance generation and solving times however vary significantly, costs which can no longer be justified. We believe this result is a consequence of the design of our instance generation framework. By using such a tailored approach and not relying on uniform sample distributions for all areas, it appears we are able to capture more knowledge about the study case and reach a stable level of performance across all configurations. The reader will most likely note that the effective rates are relatively low, with a 75% maximum when the model guarantees 100%. This gap is due to missing OSM data and agent restrictions, on which we will elaborate in the next section.

III.3 Relaxing the full-coverage constraint

The second hypothesis brought forward in Chapter II is that the enforcement of a full coverage constraint for our application leads to a significant increase in computational cost but does not proportionately improve the quality of the corresponding CCTV infrastructure. In other words, it appears possible to considerably reduce cost by allowing for relatively minor tolerance margins in the theoretical (SCP) solution.

In order to explore this hypothesis, we propose the following approach. For each instance, we first generate a front of solutions which provide varying levels of coverage, from a single camera being set up to the use of the complete full coverage greedy SCP solution. To generate these solutions, we need to introduce another yet similar combinatorial problem: maximum k-set cover (MkCP).

Problem (maximum cover)**.** *Given an integer k, a set of elements $\mathcal{I}$ (rows), a collection of sets $\mathcal{J}$ (columns) such that the union of all sets in $\mathcal{J}$ is $\mathcal{I}$, find the subset $\mathcal{C} \subset \mathcal{J}$ with $|\mathcal{C}| = k$ such that $|\bigcup_{e \in \mathcal{C}} e|$ is maximised. In other words, identify*

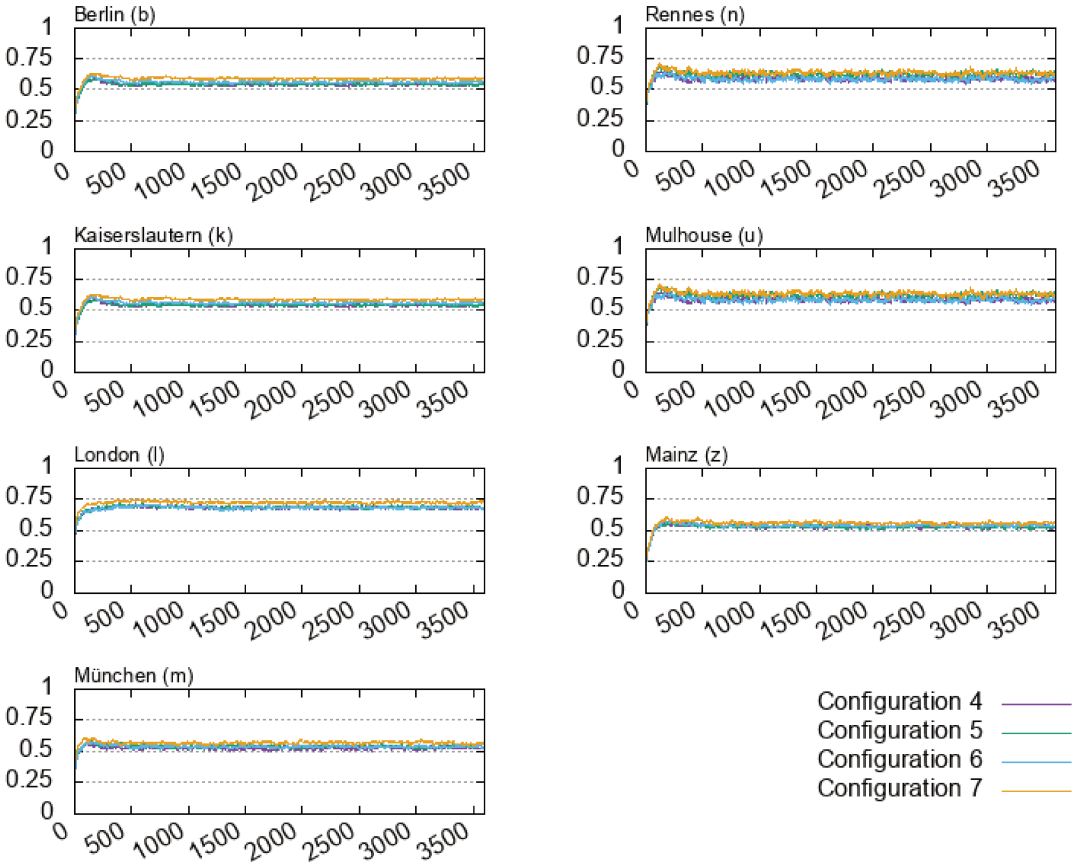


Figure III.3: Effective coverage rates when using full coverage solutions in simulations

the k-subset of $\mathcal{J}$ which covers the most elements from $\mathcal{I}$. By assigning a gain to each row, the problem can also be that of finding the subset with maximum gain.

While the search for a somewhat universally efficient algorithm for the SCP is still ongoing, research on the MkCP now considers the greedy algorithm (Algorithm 5) as the best possible polynomial-time approximation of the problem with a factor guarantee (from optimality) of $1 - \frac{1}{e}$ [70]. The problem therefore appears to be a good candidate for a tool to navigate partial solutions for the OCP efficiently. As a quick preview of the algorithm's performance for our problem, Table III.3 reports on its theoretical coverage rates when k is set to the best-known full-coverage (SCP) solution (Table III.2). In the worst-case (instance `k5`), the algorithm still covers 98.3% of the samples and terminates under 30 seconds on 2.5-2.67GHz Intel

processors.

Algorithm 5 The greedy algorithm for the maximum k-set covering problem

```
1: procedure MkCP-Greedy(I, J, k)
2:     z ← an empty solution
3:     i ← 0
4:     while i < k and |J \ z| > 0 do
5:         j ← arg max_{j'∈(J\z)} |j' \ ∪_{j''∈z} j''|
6:         z ← z ∪ {j}
7:         i ← i + 1
8:     end while
9:     return z
10: end procedure
```

Cfg. \ City	b	k	l	m	n	u	z
4	1448	1500	2086	1000	392	185	742
5	1368	1403	2027	982	374	181	673
6	1408	1358	2078	1075	350	172	642
7	1649	1500	2943	1304	426	195	744

Table III.2: Best-known full-coverage (SCP) solutions to the second set of real-world instances

To compute our solution front, we begin to compute the extrema: one camera and greedy full coverage. Once these have been identified, the other solutions can be identified as follows. A number n of solutions is set which defines the accuracy of the front: more solutions help define a more precise front, but also require more computation time. If z is the full coverage greedy solution mentioned earlier and the front is to hold n solutions, the front's frequency is defined as $f = \frac{|z|-1}{n}$. The first point is therefore an MkCP solution with $k = 1 + f$. The other points are defined incrementally up to $k = |z| - f$ and the associated solutions are computed using Algorithm 5.

Figure III.4 draws the fronts defined above, one plot per city. The x-axis represents the number of cameras while the y-axis is a measure of the theoretical coverage rate. The vertical lines are the best known full coverage solutions reported in Table III.2. The squares are the curve knees, defined as the points of strongest deviation and computed using the Kneedle algorithm designed by Satopaa et al. [169].

Cfg. \ City	b	k	l	m	n	u	z
4	0.988	0.987	0.991	0.992	0.985	0.986	0.990
5	0.988	**0.983**	0.991	0.993	0.987	0.986	0.988
6	0.993	0.988	0.995	0.997	0.988	0.985	0.989
7	0.997	0.993	0.997	**0.998**	0.996	0.994	0.995

Table III.3: MkCP greedy coverage rates when setting k to the best known SCP solution value

We chose to highlight these values because they represent clearly-defined breaking points beyond which decision makers should expect their coverage rates to drop more significantly. In other words, the section of a front on the right-hand side of its knee can be consider as room for cost/coverage trade-offs. As Figure III.4 shows, this section spans across the majority of the cost range.

Figure III.4 suggests that there is indeed a significant margin available for cost-coverage trade-offs. As of right now however, all measures have relied on the SCP model: the (theoretical) coverage rates are based on the number of samples, which have to be covered, at least partially, due to the model's combinatorial constraints. The shape of the front, and therefore of the trade-off range, could be different when measured outside of the scope of the SCP model. Taking Berlin as an example: the front knees roughly represent a 50% cost reduction for a 25% sample loss. The actual coverage loss may however be different if measured independently, in continuous space.

It is therefore necessary to ensure that the shape of the fronts computed here matches that of the effective coverage rate when measuring the efficiency of the corresponding CCTV networks. In order to do so, we propose a second round of simulations using the same experimental setup as before. We began by regenerating the front's solutions, but this time at regular intervals based on coverage rates: a solution at 10%, another at 20%, and so on, with a focus on the 70%-100% for which we used a 5% interval. An issue arises however: because the fronts are not linear, the number of cameras cannot help target specific coverage rates directly. To circumvent this, we propose a binary-search method to help locate solutions with specific coverage rates (Algorithm 6).

Figure III.5 reports on our results for this section, and allows for a comparison of theoretical and effective coverage rates. In other words, it enables validation of the theoretical fronts by evaluating coverage in continuous space for various MCP solutions. Note that given the conclusions reached in the previous section regarding

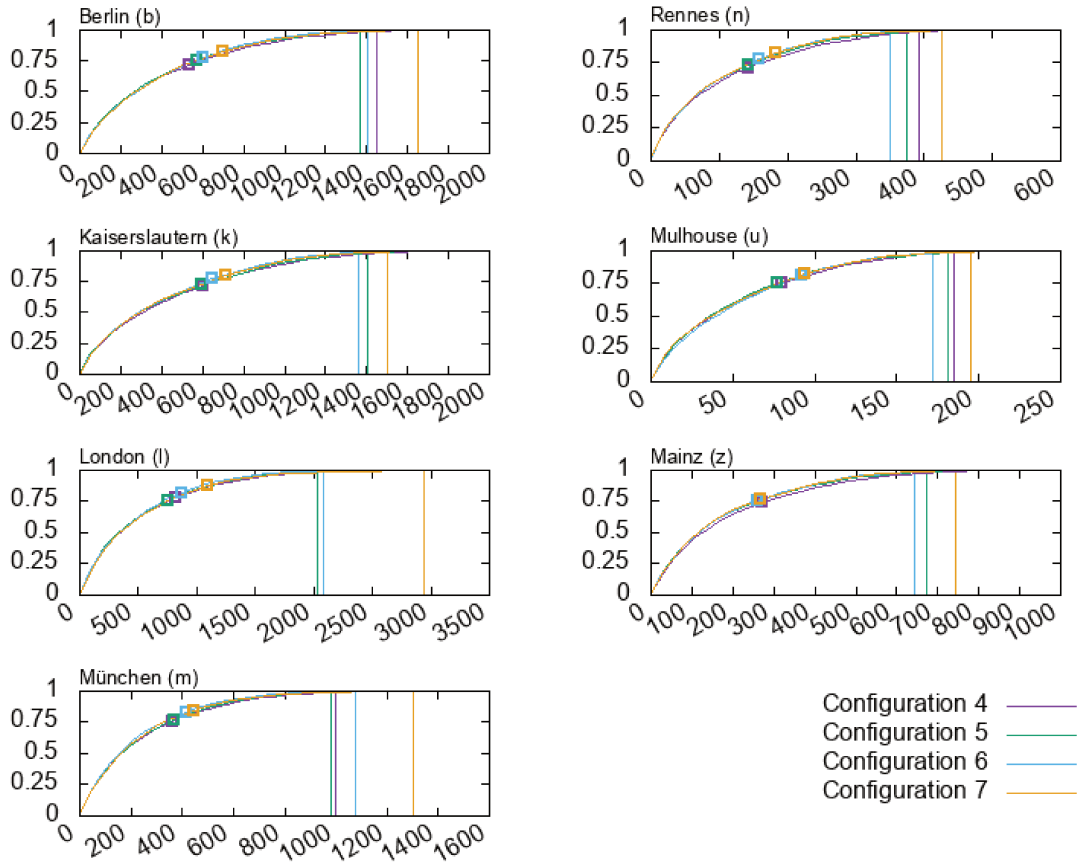


Figure III.4: MkCP solution fronts (solution costs and coverage rates)

sampling frequencies, we now only consider configuration 4.

The reported theoretical coverage rates correspond to the fronts presented earlier in Figure III.4, with the configuration 4 knee point to help delimit the expected trade-off range. The second curve is the average effective coverage rate observed during a simulation run (i.e. over 3600 steps). The x-axis still represents cost, while the y-axis keeps track of coverage, either in terms of samples (theoretical) or agents (effective).

As the reader will notice, both measures follow the same pattern, save for an offset which can be attributed to missing OSM data. Indeed, in continuous space, agents are allowed to wander into open areas which were never sampled as they correspond to undocumented private lots. In cities where OSM contributors are more numerous (e.g. London) or where most of the open space is public, the curves remain closer

Algorithm 6 The binary search algorithm for MkCP solutions at given coverage rates

procedure MCP-Binary-Search($\mathcal{I}, \mathcal{J}, r$)
  $z^+ \leftarrow$SCP-Greedy($\mathcal{I}, \mathcal{J}$)
  $z^- \leftarrow$MCP-Greedy($\mathcal{I}, \mathcal{J}, 1$)
  while $|z^+| \neq |z^-|$ **do**
    $k \leftarrow \left\lceil \frac{|z^+| + |z^-|}{2} \right\rceil$
    $z' \leftarrow$MCP-Greedy($\mathcal{I}, \mathcal{J}, k$)
    $r' \leftarrow \frac{|\cup_{j \in z'} j|}{|\mathcal{I}|}$
    if $r' < r$ **then**
      $z^- \leftarrow z'$
    else
      $z^+ \leftarrow z'$
    end if
  end while
  return z^+
end procedure

together.

Should these gaps in the available data be filled, the simulations make it safe to assume that the OCP/MkCP models are indeed suited to our application. The theoretical fronts, which remain easy to compute, faithfully mirror the data recovered from simulations in continuous space. They can therefore be used by decision makers to balance cost and coverage. Knee points can help identify the range in which to negotiate costs, and the efficient MkCP greedy algorithm can be used to yield appropriate solutions. This framework effectively provides users with good and applicable solutions and allows the optimality requirement to be dropped when applying such models to the area coverage problems in urban areas.

Review

This chapter covered three sets of related experiments. The first was a numerical benchmark of the state of the art for OCP and applied SCP literature, a study which, to the best of our knowledge, had not been performed before due to the large variety of applications and problem variants. Our results showed that instances

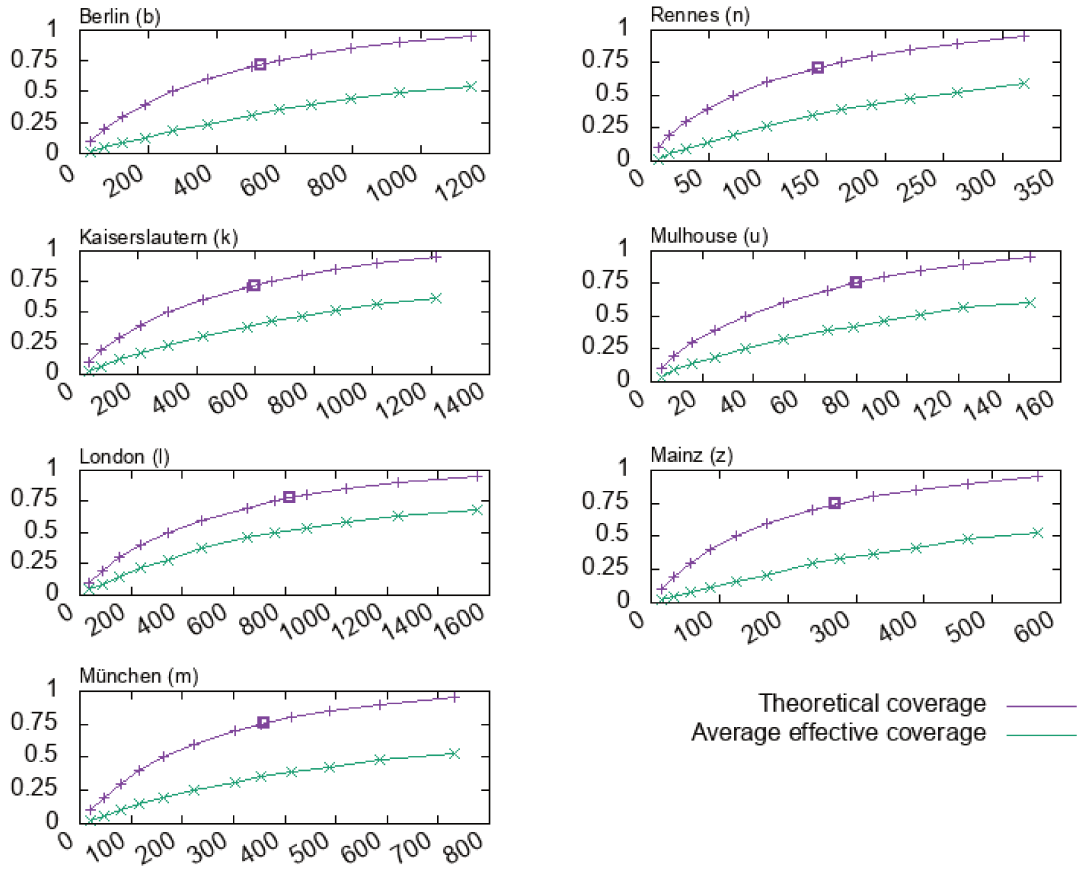


Figure III.5: Theoretical and average effective coverage rates in configuration 4

generated with our framework, introduced in Chapter II, are generally easier to solve. Larger instances however still allowed us to categorise the approaches found in the literature and identify valuable algorithmic components to be used in Chapter IV. These experiments also suggests several hypotheses regarding sampling frequencies and the impact of full coverage constraints. The next two sets of experiments were aimed at confirming those hypotheses and indeed confirmed them: high-resolution sampling frequencies and full coverage constraints do not lead to a boost in the quality of the corresponding CCTV networks, and therefore do not justify the associated computational cost increase. Our efforts should therefore turn to approaches which help users better control and guide coverage trade-offs. Such an approach is presented in Chapter IV.

Symbol	Cfg.	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{J}\rvert$	Density	Generation time
b	4	45797	14288	0.0011	00h 11m 05.79s
	5	66913	25562	0.0011	00h 25m 56.08s
	6	269158	51836	0.0010	08h 25m 50.67s
	7	460703	79899	0.0010	17h 55m 59.42s
k	4	42679	13114	0.0010	00h 03m 09.53s
	5	61394	24337	0.0010	00h 13m 38.34s
	6	247730	55091	0.0009	06h 40m 01.66s
	7	423524	84145	0.0009	07h 06m 51.65s
l	4	60230	17002	0.0008	00h 05m 58.58s
	5	87792	32241	0.0008	00h 21m 13.65s
	6	334519	74494	0.0007	06h 02m 36.11s
	7	588612	109137	0.0007	08h 41m 47.76s
m	4	33169	8915	0.0017	00h 03m 14.80s
	5	48285	16206	0.0017	00h 08m 28.35s
	6	191366	38927	0.0015	03h 36m 47.60s
	7	329800	55871	0.0015	07h 09m 46.58s
n	4	12561	3436	0.0044	00h 02m 01.06s
	5	18242	6244	0.0044	00h 02m 51.24s
	6	70279	13958	0.0042	01h 03m 50.87s
	7	120786	19430	0.0039	03h 17m 07.06s
u	4	4881	1424	0.0083	00h 00m 55.96s
	5	7245	2810	0.0081	00h 01m 24.49s
	6	25700	6650	0.0070	00h 27m 21.06s
	7	45691	9458	0.0069	01h 23m 12.48s
z	4	27390	6102	0.0021	00h 02m 02.61s
	5	39410	11286	0.0022	00h 02m 40.98s
	6	156439	22550	0.0022	01h 10m 26.82s
	7	268640	32692	0.0021	01h 57m 32.43s

Table III.4: Basic statistics for our second set of 28 real-world instances

Chapter IV

Human-assisted optimisation

Introduction

Concluding on the work done so far, we now move on to the final stage of this thesis and propose both a model and a solving implementation to tackle optimal camera placement in the context of decision support systems. More specifically, we focus on a use case in which users are invited to regularly provide input based on their own appreciation of the current solution. The algorithms are therefore expected to run and react quickly to user input to provide solution updates at a pace which maintains interactability. We make use of the conclusions reached in Chapter III in order to propose a mixed model which brings together the two combinatorial problems discussed so far. The instance generation framework introduced in Chapter II is also adjusted to better suit our specific application. Finally, we make use of our observations from Chapter III regarding coverage constraints and propose a greedy-based solving algorithm which interprets user input into ground sample weights for targeted coverage.

Related publication

[121] Julien Kritter et al. "On the use of human-assisted optimisation for the optimal camera placement problem and the surveillance of urban events". In: *7th 2020 International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, June 2020

IV.1 Model, application-specific constraints and preprocessing

Before we can proceed to effectively solving the optimal camera placement problem, several practical aspects related to our application and decision support systems must be taken into consideration and integrated into our model and approach.

The first element to consider is the budget. So far, we have mostly addressed the problem using a standard set covering model with full coverage constraints. This leaves the cost of the infrastructure loose, which means there are no restrictions as to how much can be invested in CCTV infrastructure. Unfortunately for some, however, money and human labour are two limited resources and one cannot simply ask a city or region to set up an arbitrarily expensive network of cameras. On this aspect, the use of the maximum k-set covering model introduced in Chapter III seems to be more appropriate.

Removing the full coverage constraint does however create a new problem: if coverage also becomes a limited resource, one must decide which parts of the surveillance areas are not important enough to be allocated cameras. This is where another aspect of our use case needs to be considered. The goal of our work is to provide an approach for decision support systems. In other words, our algorithms are not meant to be run once and yield a final non-debatable solution. The availability of a user for input should definitely be exploited. To this end, it is necessary to come up with an approach which takes in input for the user and translates it into a weighting system through which the solving algorithm can decide where to focus coverage and which areas can be left in the dark. Provided that the algorithms run quickly enough, this would enable users (law enforcement agencies, town officials, ...) to continuously review solutions and provide feedback iteratively until the proposed infrastructure meets their needs. This is particularly well suited to our application as it would be very difficult to acquire and formalise such knowledge from standard datasets, which simply cannot capture the added value of field experience and predictions made by officers regarding the upcoming urban events.

While the trade-off approach introduced in Chapter III does seem to meet the above requirements, it should be noted that some parts of a surveillance area simply cannot be left uncovered. For such cases, full coverage constraints still need to be enforced. The solving approach should therefore allow the user to specify so-called critical areas for which the typically full coverage formulation is to be used. To combine both the weighting scheme and the critical areas, we propose the following model, which combines the SCP and the MkCP. The notations used are explained

in Table IV.1.

$$\textbf{max} \quad \sum_{i=1}^{|\mathcal{I}|} w_i y_i \tag{IV.1}$$

$$\sum_{j=1}^{|\mathcal{J}|} x_j \quad \leq k \tag{IV.2}$$

$$\sum_{j=1}^{|\mathcal{J}|} a_{ij} x_j \quad \geq y_i \qquad \forall i, 1 \leq i \leq |\mathcal{I}| \tag{IV.3}$$

$$y_i \quad \geq r_i \qquad \forall i, 1 \leq i \leq |\mathcal{I}| \tag{IV.4}$$

Notation	Description
w_i	Weight of sample i
y_i	Decision variable for sample i's coverage
x_j	Decision variable for candidate j
k	Overall budget
a_{ij}	Entries in the binary visibility matrix
r_i	Critical status for sample i

Table IV.1: Notation summary for the mixed SCP-MkCP model

In this new model, Equation (IV.1) is a typical MkCP coverage maximisation objective. Equation (IV.2) enforces the budget while (IV.4) ensures full coverage of the critical areas. Equation IV.3 binds the two sets of decision variables together.

Solving aside, other practical elements must also be taken into account and can easily be integrated into the instance generation and preprocessing framework presented in Chapter II. When creating an instance, it should be possible to force subareas to remain in the dark: any camera candidate which covers a sample in those areas should be discarded. A common example in France as to when this is necessary are residential buildings: windows into private lots should not be covered as this would go against national privacy laws [122]. Another example are embassies and consulates which are to be considered as foreign territory and for which surveillance is typically left to the associated nation. On the other hand, it should also be possible to cover important structures and buildings, and not just the ground. This typically applies to popular monuments, administrative buildings and so on. These can also be subject to specific rules which prevent their use for camera mounting, in which case they should be excluded from candidate sampling. All these constraints

require us to review the way we categorise map elements: samples can no longer be associated to the ground (horizontal surfaces). Similarly, we can also detach cameras from walls if we allow them to be placed on non-vertical surfaces like porches. More generally, we will now categorise samplables (map elements which provide samples) using 4 questions:

1. Should we ignore (no sampling), cover or hide the element?
2. Should we sample horizontally (road, roof, open ground area, ...) or vertically (pole, wall) ?
3. Will we work on a single node, a line or a polygon?
4. Will we be sampling targets (former **ground samples**) or candidates?

The combined answers to these questions give the category of an element. For example, a horizontal line target samplable is a road to be covered. A vertical node candidate samplable is an electric pole or traffic light on which cameras can be set up. A horizontal polygon candidate samplable is a porch or any kind of roof to the underside of which cameras can be attached. Only two combinations are ignored: horizontal node candidate and horizontal line candidate, although they could certainly be conceived in particularly elaborate schemes. Note that the answer to the first question does not really affect the method used for sampling, only the meaning of the resulting samples (if any). For all the others combinations, the sampling algorithm are either known (see Chapter II) or easily derived for our previous approaches (for example, sampling porches combines the strategies for structure points and ground polygons, which were both introduced earlier). It should be noted that the categorisation is not actually part of our code base: users get to decide how each element is categorised and processed through filter rules. This allows us to fine-tune the sampling process based on local regulations and practices.

Finally, we integrated our earlier conclusions on sampling frequencies and replaced all sampling parameters with a single integer value which we named rigour. A rigour value of 1 corresponds to configuration 4 (1 for target sampling) from Chapter II. Higher values yield finer sampling frequencies in a linear fashion which we summarise in Table IV.2.

IV.2 User interactions and assisted solving

Based on our results from Chapter III, we propose to use a combination of two greedy algorithms, primarily motivated by their demonstrated efficiency of the MkCP

Parameter	f_g^l	f_g^w	f_a^g	a_g^{min}	a_g^{max}	f_s^l	f_s^a	a_s^{min}	a_s^{max}	f_s^p	f_s^t
Value	$\frac{10}{r}$	$\frac{10}{r}$	$\frac{10}{r}$	0	3	$\frac{10}{r}$	$\frac{10}{r}$	3.5	5.5	$\frac{\pi}{5+r}$	$\frac{\pi}{5+r}$

Table IV.2: Parameter values derived from the general rigour value r

(constraint (IV.2), see Table III.3). The first stage of the algorithm aims at fulfilling constraints (IV.4): the critical areas registered by the user are therefore covered first. The use of a greedy algorithm at this stage has two advantages: it is fast, and it allows us to insert a look-ahead bias for the second phase of our approach. This is achieved by breaking ties favouring candidates which offer the best coverage on nearby non-critical targets. As it so happens, tie-breaking tends to happen very often when solving an SCP subproblem, an observation which coincides with our conclusions from Chapter III). The reader might argue that using a greedy algorithm at this stage represents a significant loss in solution quality. Given that we are typically working with very small critical subsets however (wrt. the size of the instance), we observe that the time savings are certainly worth this trade-off. Besides, as was shown in our earlier benchmark (see Figure III.1), greedy algorithms are not significantly outperformed on smaller problems.

The second stage of our algorithm is an implementation of the greedy heuristic for the MkCP. We chose to structure the process in a target-oriented fashion, which is slightly different from the original implementation which ranks candidates and picks them in order. This decision was made after we observed a rather unfair bias which caused candidates with large covers to severely overpower our weighting scheme and led to less favourable outcomes in larger surveillance areas. The overall algorithm is illustrated in Algorithm 8. The column redundancy checks are simple procedures which remove candidates for which all associated samples are covered by more than one candidate in the solution. Both these checks and the greedy components run in polynomial time, rendering the overall algorithm very efficient and therefore suitable for our use case.

We implemented our algorithm as part of our initial code base, which we then extended to include a user interface. As we have mentioned earlier, our algorithm's efficiency can only truly be exploited if user input is regularly provided. This is done through a map-based interface which communicates with our higher-performance code in order to request solutions based on user input and render them. Figure IV.1 gives an overview of the interface on a use case in the city of Strasbourg, France.

The solving process typically goes as follows. First, project specifications are

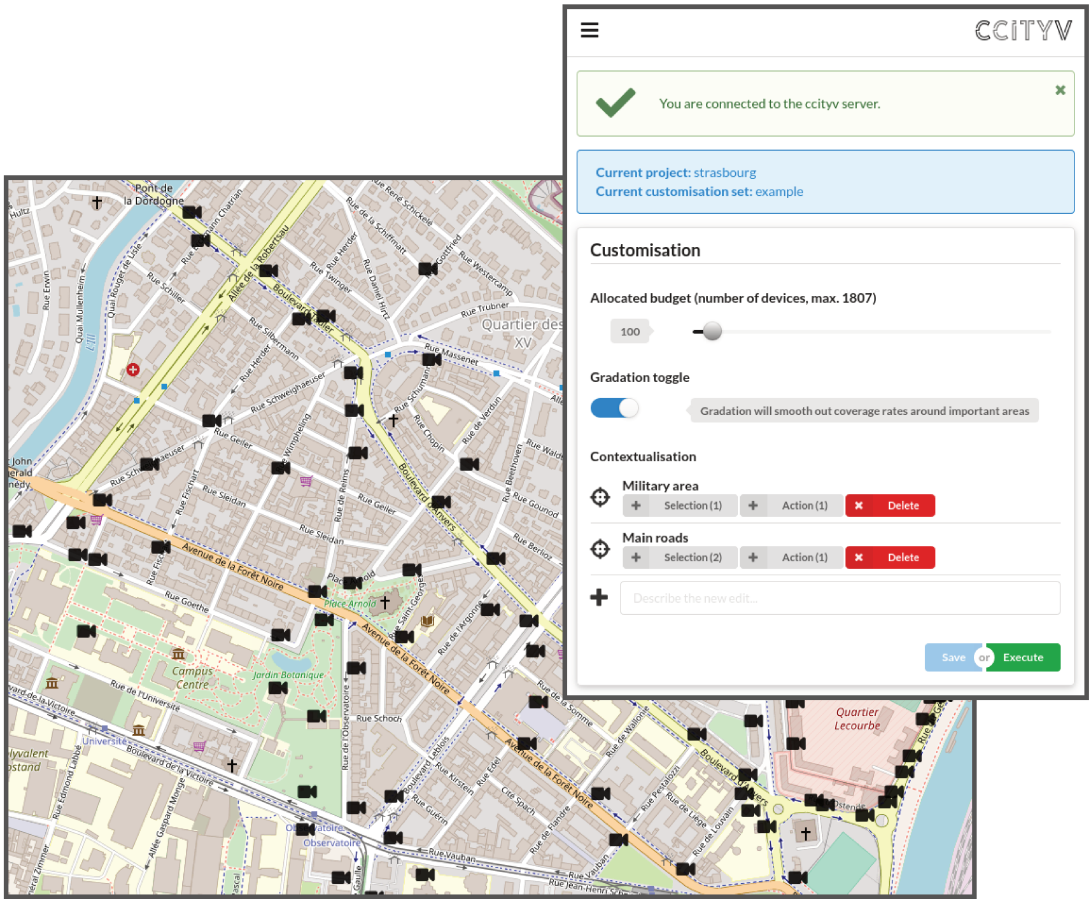


Figure IV.1: Basic user interface components for decision support systems

provided and define the surveillance area as well as the various filtering rules used for sampling and occlusion detection. Instance generation may then begin: the area is sampled, visibility analysis is performed, the instance is reduced and stored. The user can then log into the interface and open the project. The map will appear along with a panel used for input. The user may then set a budget limit, which we cap at the greedy SCP solution value. Contextualisation may then be provided: routes and areas can be weighted using arbitrary integers, or set as critical. These parameters can then be sent over to the solver which responds with a set of camera setup directives. The associated locations are then rendered on the map and the user can review their settings to adjust specific areas. The process continues iteratively until the user is satisfied with the solution.

An important note should be made regarding user-provided weights. It is clear from our problem formulation that target weights are bound to have a strong influence over the solving algorithm's decisions. These weights are provided by the user and lie within the [1; 100] range. While this indeed allows the user to guide the algorithm through frequent weight adjustments, it does so in a very abrupt fashion, leaving weighted areas as soon as they are covered and quickly moving on to candidates with larger covers. While this might be the desired result when weights are very finely defined, it would make much more sense for our application if the areas surrounding weighted ones would have some importance as well. For this reason, we have made it possible to have weights propagated across the area before solving. This process is illustrated in Algorithm 7 and Figure IV.2, in which $d(i, i')$ is used to denote the Euclidean distance between targets i and i'. The process simply computes a new weight for every target based on its distance from the nearest user-weighted target. This effectively means that targets which are close to weighted areas gain weight and that this gain weakens as the targets get further away. It also lessens the impact of user-provided weights and prevents the algorithm from making somewhat extreme choices.

Algorithm 7 Weight gradation algorithm

procedure WEIGHT-GRADATION($\mathcal{I}$, $\mathcal{J}$, w)
  $\mathcal{W} \leftarrow \{i \mid i \in \mathcal{I}, w_i > 1\}$ ▷ user-weighted points
  $\mathcal{N} \leftarrow \{\arg\min_{i' \in \mathcal{W}} \{d(i, i')\} \mid i \in \mathcal{I}\}$
  $d_{max} \leftarrow \max\{d(i, \mathcal{N}_i) \mid i \in \mathcal{I}\}$
  for $i \in \mathcal{I} \setminus \mathcal{W}$ **do**
    $w_i \leftarrow 1 + (1 - \frac{d(i,\mathcal{N}_i)}{d_{max}})(w_{\mathcal{N}_i} - 1)$
  end for
end procedure

To illustrate our approach, we used a simple study case on a part of the city of Strasbourg, France. We defined a small set of roads as important by giving them weight, and set a military area as critical. Figure IV.3 demonstrates the impact of user-provided weight and the smoothing behaviour which can be requested from the algorithm by toggling gradation.

We observe that the base solution distributes cameras all over the area but focuses on large open spaces such as parks (Figure IV.2). This is because a lot of the candidates around these areas have large covers: there is no occlusion. Adding weights overrides this default behaviour and indeed focuses coverage on a selection of main roads (Figure IV.2). However, as was stated earlier, this behaviour is lim-

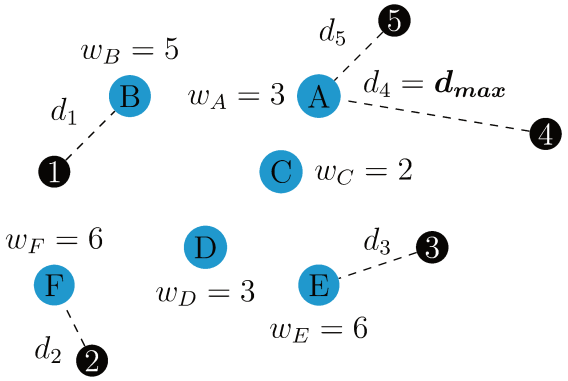
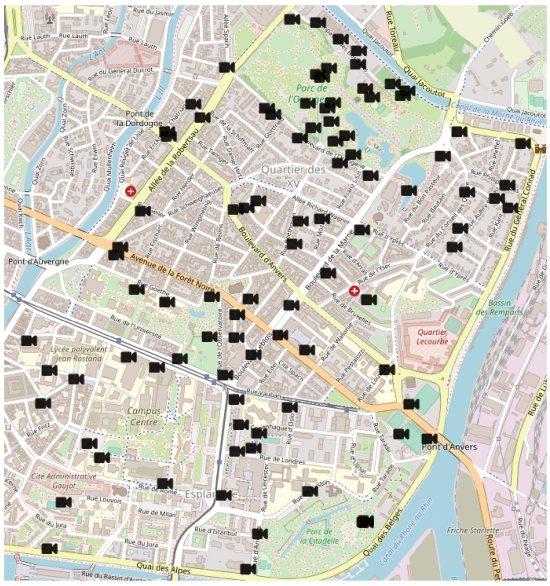


Figure IV.2: Illustration for Algorithm 7. Unweighted (black) point 4 remains at weight 1 as it is furthest away from a weighted (blue) point (A). Unweighted point 1 is assigned $w_1 = 1 + 4\frac{d_i}{d_{max}}$, which tends towards $w_B = 5$ as $d(1, B)$ decreases.

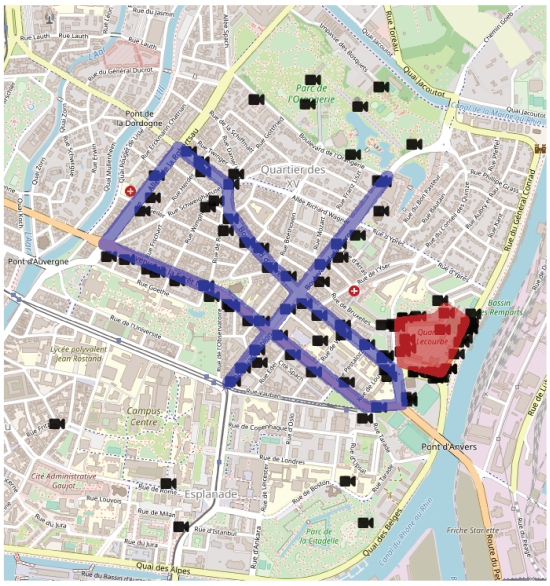
ited and the algorithm returns to large covers quickly. This is compensated for using gradation: weights are balanced across the area and elements surrounding the main roads (crossroads, endpoints, nearby paths, ...) are now more densely covered (Figure IV.2).

Review

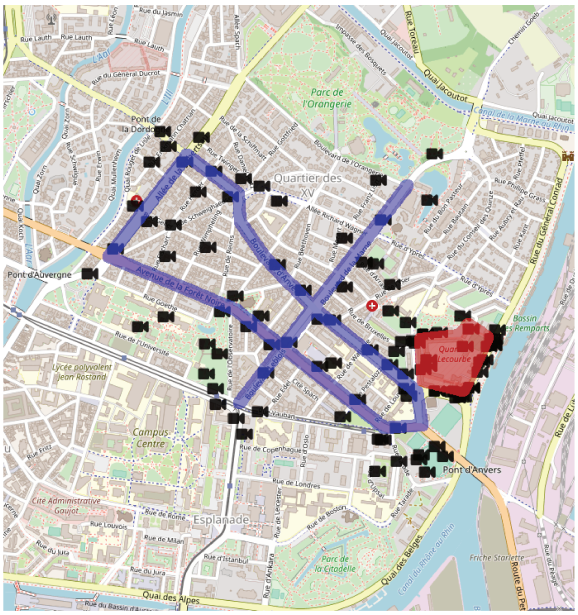
This chapter has brought together the hypotheses and conclusions tackled in earlier work. We introduced final changes to our instance generation procedure, taking into account additional application-specific needs as well as the upcoming integration into a decision support system. Most of the parameters of the sampling procedures have effectively been removed following our conclusions from Chapter III. Based on those same results, we then proposed a greedy-based solving approach. The algorithm is able to take into account user input and runs quickly enough so as not to impede user interaction. A solving process is therefore designed through which the user is able to visualise the incumbent solution on a map and provide new weight adjustments iteratively until the solution is deemed satisfactory. We presented the impact of all the above components using a small study case in the city of Strasbourg, France.



(a) Base solution



(b) Setting weights (blue) and a critical area (red)



(c) Toggling weights gradation

Figure IV.3: Example solutions

Algorithm 8 A greedy-based solving algorithm for our SCP-MkCP model

procedure SCP-MkCP-Solve($\mathcal{I}$,$\mathcal{J}$, w)
  $crit \leftarrow$ a zero vector of $|\mathcal{J}|$ elements
  $ncrit \leftarrow$ a zero vector of $|\mathcal{J}|$ elements
  for $i \in \mathcal{I}$ **do**
    for $j \in \mathcal{J}_i$ **do**
      if r_i **then**
        $crit_j \leftarrow crit_j + 1$
      else
        $ncrit_j \leftarrow ncrit_j + w_i$
      end if
    end for
  end for
  $z \leftarrow \emptyset$
  while $|z| < k$ and $\max\{crit\} \neq 0$ **do**
    $\mathcal{J}' \leftarrow \{j \in \mathcal{J} \mid j \notin z, crit_j = \max\{crit\}\}$
    $\mathcal{J}'' \leftarrow \{j \in \mathcal{J}', ncrit_j = \max\{ncrit\}\}$
    $j \leftarrow$ a random candidate from $\mathcal{J}''$
    $z \leftarrow z \cup \{j\}$
    for $i \in \mathcal{I}_j$ **do**
      for $j \in \mathcal{J}_i \setminus z$ **do**
        if r_i **then**
          $crit_j \leftarrow crit_j - 1$
        else
          $ncrit_j \leftarrow ncrit_j - w_i$
        end if
      end for
    end for
    prune redundant columns created by j in z
  end while
  while $|z| < k$ and $\cup_{j \in z}\{\mathcal{I}_j\} \neq \mathcal{I}$ **do**
    $\mathcal{I}' \leftarrow \{i \in \mathcal{I} \mid \mathcal{J}_i \cup z = \emptyset\}$
    $\mathcal{I}'' \leftarrow \{i \in \mathcal{I}' \mid w_i = \max_{i' \in \mathcal{I}'}\{w_{i'}\}\}$
    $i \leftarrow$ a random sample in $\mathcal{I}''$
    $j \leftarrow \arg\max_{j' \in \mathcal{J}_i}\{ncrit_{j'}\}$
    $z \leftarrow z \cup \{j\}$
    for $i \in \mathcal{I}_j$ **do**
      for $j \in \mathcal{J}_i \setminus z$ **do**
        $ncrit_j \leftarrow ncrit_j - w_i$
      end for
    end for
    prune redundant columns created by j in z
  end while
end procedure

Conclusion

In addition to the individual chapter reviews found throughout this report, we may now take time to look back at our research work, evaluate its overall suitability with regards to our project objectives and highlight its shortcomings as possible open lines of research.

Our literature review had as its main objective to bring forward a relationship between an application, optimal camera placement, and a popular combinatorial problem, set cover. So far, authors had chosen to set that relationship aside and the two subjects of research were mostly left isolated. It is our hope that our survey has successfully highlighted this discrepancy and may encourage further optimal camera placement research to more regularly integrate work done on the set cover problem. This in turn may prove to be of interest for the latter: as we have seen in Chapters II and III, applications may yield specific instances which could be further studied from a combinatorial standpoint. Aside from this, we believe the review may serve as a useful reference about the various models and algorithmic components which have been used, more or less successfully on our problem and its variants.

Chapter II has addressed problem modelling, with a focus on reconciling the extremes found in the literature. We proposed an instance generation and preprocessing framework capable of extracting an accurate representation of any publicly documented area, and generating the associated combinatorial instances through efficient visibility analysis and reduction algorithms. We currently believe that the level of realism granted by our approach is sufficient for our specific application. One can however probably find many ways to further improve the process, for example by using more advanced building model (BIM) databases.

By selecting a few study cases, we were later able to establish a clear baseline for the state of the art. In Chapter III, we performed such a study and highlighted new hypotheses about the complexity of our problem. It would undoubtedly be interesting to further study our instances to better isolate the elements which most significantly impact complexity.

Our conclusions on the aforementioned hypotheses finally led us to design a human-assisted optimisation approach which not only exploits the trade-offs made possible in the previous chapter, but also addresses the problem in a user-centric fashion such that input and feedback may continuously be integrated into solutions. The approach is consistent with the specifications of a decision support system, an example of which we also presented. Again, several improvements could still be considered, in particular the use of existing use-case-specific datasets to extract more knowledge about the city's layout and habits. Better integration may also be achieved by taking into account existing CCTV infrastructure and therefore working more extensively with several camera models at the same time. Finally, as part of a decision support system, our approach may find ways to benefit from other components, for example by taking account real-time crowd behaviour data or the availability of additional support provided by officers on site or devices such as drones.

Bibliography

[1] José H. Ablanedo-Rosas and César Rego. "Surrogate constraint normalization for the set covering problem". In: *European Journal of Operational Research* 205.3 (Sept. 2010), pp. 540–551. DOI: 10.1016/j.ejor.2010.02.008. URL: https://doi.org/10.1016%2Fj.ejor.2010.02.008.

[2] Ali Ahmadzadeh et al. "An Optimization-Based Approach to Time-Critical Cooperative Surveillance and Coverage with UAVs". In: *Experimental Robotics*. Springer Berlin Heidelberg, 2008, pp. 491–500. DOI: 10.1007/978-3-540-77457-0_46. URL: https://doi.org/10.1007%2F978-3-540-77457-0_46.

[3] Uwe Aickelin. "An indirect genetic algorithm for set covering problems". In: *Journal of the Operational Research Society* 53.10 (Oct. 2002), pp. 1118–1126. DOI: 10.1057/palgrave.jors.2601317. URL: https://doi.org/10.1057%2Fpalgrave.jors.2601317.

[4] Tycho Andersen and Srikanta Tirthapura. "Wireless sensor deployment for 3D coverage with constraints". In: *2009 Sixth International Conference on Networked Sensing Systems (INSS)*. IEEE, June 2009. DOI: 10.1109/inss.2009.5409946. URL: https://doi.org/10.1109%2Finss.2009.5409946.

[5] Franck Angella, Livier Reithler, and Frederic Gallesio. "Optimal deployment of cameras for video surveillance systems". In: *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*. IEEE, Sept. 2007. DOI: 10.1109/avss.2007.4425342. URL: https://doi.org/10.1109%2Favss.2007.4425342.

[6] S. Raja Balachandar and K. Kannan. "A meta-heuristic algorithm for set covering problem based on gravity". In: *International Journal of Computational and Mathematical Sciences* 4.5 (2010), pp. 223–228.

[7] S. Balaji and N. Revathi. "A new approach for solving set covering problem using jumping particle swarm optimization method". In: *Natural Computing* 15.3 (July 2015), pp. 503–517. DOI: 10.1007/s11047-015-9509-2. URL: https://doi.org/10.1007%2Fs11047-015-9509-2.

[8] Egon Balas and Maria C. Carrera. "A Dynamic Subgradient-Based Branch-and-Bound Procedure for Set Covering". In: *Operations Research* 44.6 (Dec. 1996), pp. 875–890. DOI: 10.1287/opre.44.6.875. URL: https://doi.org/10.1287%2Fopre.44.6.875.

[9] Egon Balas and Andrew Ho. "Set covering algorithms using cutting planes, heuristics, and subgradient optimization: A computational study". In: *Mathematical Programming Studies*. Springer Berlin Heidelberg, 1980, pp. 37–60. DOI: 10.1007/bfb0120886. URL: https://doi.org/10.1007%2Fbfb0120886.

[10] Michael Baum and Kevin Passino. "A Search-Theoretic Approach to Cooperative Control for Uninhabited Air Vehicles". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug. 2002. DOI: 10.2514/6.2002-4589. URL: https://doi.org/10.2514%2F6.2002-4589.

[11] Joaquín Bautista and Jordi Pereira. "A GRASP algorithm to solve the unicost set covering problem". In: *Computers & Operations Research* 34.10 (Oct. 2007), pp. 3162–3173. DOI: 10.1016/j.cor.2005.11.026. URL: https://doi.org/10.1016%2Fj.cor.2005.11.026.

[12] John E. Beasley. "An algorithm for set covering problem". In: *European Journal of Operational Research* 31.1 (July 1987), pp. 85–93. DOI: 10.1016/0377-2217(87)90141-x. URL: https://doi.org/10.1016%2F0377-2217%2887%2990141-x.

[13] John E. Beasley. "OR-Library: Distributing Test Problems by Electronic Mail". In: *Journal of the Operational Research Society* 41.11 (Nov. 1990), pp. 1069–1072. DOI: 10.1057/jors.1990.166. URL: https://doi.org/10.1057%2Fjors.1990.166.

[14] John E. Beasley and Paul C. Chu. "A genetic algorithm for the set covering problem". In: *European Journal of Operational Research* 94.2 (Oct. 1996), pp. 392–404. DOI: 10.1016/0377-2217(95)00159-x. URL: https://doi.org/10.1016%2F0377-2217%2895%2900159-x.

[15] John E. Beasley and K. Jørnsten. "Enhancing an algorithm for set covering problems". In: *European Journal of Operational Research* 58.2 (Apr. 1992), pp. 293–300. DOI: 10.1016/0377-2217(92)90215-u. URL: https://doi.org/10.1016%2F0377-2217%2892%2990215-u.

[16] Norbert Beckmann et al. "The R^*-tree: an efficient and robust access method for points and rectangles". In: *Proceedings of the 1990 ACM SIGMOD international conference on Management of data - SIGMOD '90*. ACM Press, 1990. DOI: 10.1145/93597.98741. URL: https://doi.org/10.1145%2F93597.98741.

[17] Michel Berkelaar, Kjell Eikland, and Peter Notebaert. *LPSolve*. 2004.

[18] Nehme Bilal, Philippe Galinier, and Francois Guibault. "A New Formulation of the Set Covering Problem for Metaheuristic Approaches". In: *ISRN Operations Research* 2013 (2013), pp. 1–10. DOI: 10.1155/2013/203032. URL: https://doi.org/10.1155%2F2013%2F203032.

[19] Robert Bodor, Paul Schrater, and Nikolaos Papanikolopoulos. "Multi-camera positioning to optimize task observability". In: *Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.* IEEE, 2005. DOI: 10.1109/avss.2005.1577328. URL: https://doi.org/10.1109%2Favss.2005.1577328.

[20] Robert Bodor et al. "Optimal Camera Placement for Automated Surveillance Tasks". In: *Journal of Intelligent and Robotic Systems* 50.3 (Oct. 2007), pp. 257–295. DOI: 10.1007/s10846-007-9164-7. URL: https://doi.org/10.1007%2Fs10846-007-9164-7.

[21] *Boost.Geometry, spatial indices, r-trees*. https://www.boost.org/doc/libs/1_63_0/libs/geometry/doc/html/geometry/spatial_indexes/introduction.html. Accessed: 2020-08-04.

[22] Prosenjit Bose et al. "The Floodlight Problem". In: *International Journal of Computational Geometry & Applications* 07.01n02 (Feb. 1997), pp. 153–163. DOI: 10.1142/s0218195997000090. URL: https://doi.org/10.1142%2Fs0218195997000090.

[23] B. R. Bowring. "NOTES ON THE CURVATURE IN THE PRIME VERTICAL SECTION". In: *Survey Review* 29.226 (Oct. 1987), pp. 195–196. DOI: 10.1179/sre.1987.29.226.195. URL: https://doi.org/10.1179%2Fsre.1987.29.226.195.

[24] Mathieu Brévilliers et al. "Hybrid differential evolution algorithms for the optimal camera placement problem". In: *Journal of Systems and Information Technology* (Nov. 2018). DOI: 10.1108/jsit-09-2017-0081.

[25] Mathieu Brévilliers et al. "Parallel Preprocessing for the Optimal Camera Placement Problem". In: *International Journal of Modeling and Optimization* 8.1 (Feb. 2018), pp. 33–40. DOI: 10.7763/ijmo.2018.v8.621.

[26] Stephen P. Brooks, Nial Friel, and Ruth King. "Classical model selection via simulated annealing". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 65.2 (May 2003), pp. 503–520. DOI: 10.1111/1467-9868.00399. URL: https://doi.org/10.1111%2F1467-9868.00399.

[27] Michael J. Brusco, Larry W. Jacobs, and Gary M. Thompson. In: *Annals of Operations Research* 86 (1999), pp. 611–627. DOI: 10.1023/a:1018900128545. URL: https://doi.org/10.1023%2Fa%3A1018900128545.

[28] Jesús Capitán, Luis Merino, and Aníbal Ollero. "Cooperative Decision-Making Under Uncertainties for Multi-Target Surveillance with Multiples UAVs". In: *Journal of Intelligent & Robotic Systems* 84.1-4 (Sept. 2015), pp. 371–386. DOI: 10.1007/s10846-015-0269-0. URL: https://doi.org/10.1007%2Fs10846-015-0269-0.

[29] Alberto Caprara, Matteo Fischetti, and Paolo Toth. "A Heuristic Method for the Set Covering Problem". In: *Operations Research* 47.5 (Oct. 1999), pp. 730–743. DOI: 10.1287/opre.47.5.730. URL: https://doi.org/10.1287%2Fopre.47.5.730.

[30] Alberto Caprara, Paolo Toth, and Matteo Fischetti. "Algorithms for the Set Covering Problem". In: *Annals of Operations Research* 98.1/4 (2000), pp. 353–371. DOI: 10.1023/a:1019225027893. URL: https://doi.org/10.1023%2Fa%3A1019225027893.

[31] Svante Carlsson, Bengt J. Nilsson, and Simeon Ntafos. "Optimum guard covers and m-watchmen routes for restricted polygons". In: *Lecture Notes in Computer Science*. Springer-Verlag, 1991, pp. 367–378. DOI: 10.1007/bfb0028276. URL: https://doi.org/10.1007%2Fbfb0028276.

[32] Marco Caserta. "Tabu Search-Based Metaheuristic Algorithm for Large-scale Set Covering Problems". In: *Metaheuristics*. Springer US, 2007, pp. 43–63. DOI: 10.1007/978-0-387-71921-4_3. URL: https://doi.org/10.1007%2F978-0-387-71921-4_3.

[33] Drona Pratap Chandu. "Big Step Greedy Algorithm for Maximum K-coverage Problem". In: *CoRR* abs/1506.06163 (2015). arXiv: 1506.06163. URL: http://arxiv.org/abs/1506.06163.

[34] Peng Cheng, J. Keller, and V. Kumar. "Time-optimal UAV trajectory planning for 3D urban structure coverage". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Sept. 2008. DOI: 10.1109/iros.2008.4650988. URL: https://doi.org/10.1109%2Firos.2008.4650988.

[35] Sen-ching Cheung, Jian Zhao, and M. Venkatesh. "Efficient Object-Based Video Inpainting". In: *2006 International Conference on Image Processing*. IEEE, 2006. DOI: 10.1109/icip.2006.312432. URL: https://doi.org/10.1109%2Ficip.2006.312432.

[36] Weipang Chin and Simon Christos Ntafos. "Optimum watchman routes". In: *Proceedings of the second annual symposium on Computational geometry - SCG '86*. ACM Press, 1986. DOI: 10.1145/10515.10518. URL: https://doi.org/10.1145%2F10515.10518.

[37] Dimitrios Chrysostomou and Antonios Gasteratos. "Optimum multi-camera arrangement using a bee colony algorithm". In: *2012 IEEE International Conference on Imaging Systems and Techniques Proceedings*. IEEE, July 2012. DOI: 10.1109/ist.2012.6295580. URL: https://doi.org/10.1109%2Fist.2012.6295580.

[38] Dimitrios Chrysostomou, Georgios C. Sirakoulis, and Antonios Gasteratos. "A bio-inspired multi-camera system for dynamic crowd analysis". In: *Pattern Recognition Letters* 44 (July 2014), pp. 141–151. DOI: 10.1016/j.patrec.2013.11.020. URL: https://doi.org/10.1016%2Fj.patrec.2013.11.020.

[39] Václav Chvátal. "A combinatorial theorem in plane geometry". In: *Journal of Combinatorial Theory, Series B* 18.1 (Feb. 1975), pp. 39–41. DOI: 10.1016/0095-8956(75)90061-1. URL: https://doi.org/10.1016%2F0095-8956%2875%2990061-1.

[40] Václav Chvátal. "A Greedy Heuristic for the Set-Covering Problem". In: *Mathematics of Operations Research* 4.3 (Aug. 1979), pp. 233–235. DOI: 10.1287/moor.4.3.233. URL: https://doi.org/10.1287%2Fmoor.4.3.233.

[41] Nicola Conci and Leonardo Lizzi. "Camera placement using particle swarm optimization in visual surveillance applications". In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE, Nov. 2009. DOI: 10.1109/icip.2009.5413833. URL: https://doi.org/10.1109%2Ficip.2009.5413833.

[42] Stephen A. Cook. "The complexity of theorem-proving procedures". In: *Proceedings of the third annual ACM symposium on Theory of computing - STOC '71*. ACM Press, 1971. DOI: 10.1145/800157.805047. URL: https://doi.org/10.1145%2F800157.805047.

[43] Broderick Crawford and Carlos Castro. "Integrating Lookahead and Post Processing Procedures with ACO for Solving Set Partitioning and Covering Problems". In: *Artificial Intelligence and Soft Computing – ICAISC 2006*. Springer Berlin Heidelberg, 2006, pp. 1082–1090. DOI: 10.1007/11785231_113. URL: https://doi.org/10.1007%2F11785231_113.

[44] Broderick Crawford, Ricardo Soto, and Eric Monfroy. "Cultural Algorithms for the Set Covering Problem". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 27–34. DOI: 10.1007/978-3-642-38715-9_4. URL: https://doi.org/10.1007%2F978-3-642-38715-9_4.

[45] Broderick Crawford et al. "A Binary Cat Swarm Optimization Algorithm for the Non-Unicost Set Covering Problem". In: *Mathematical Problems in Engineering* 2015 (2015), pp. 1–8. DOI: 10.1155/2015/578541. URL: https://doi.org/10.1155%2F2015%2F578541.

[46] Broderick Crawford et al. "A Cultural Algorithm for Solving the Set Covering Problem". In: *Analysis and Design of Intelligent Systems using Soft Computing Techniques*. Springer Berlin Heidelberg, 2007, pp. 408–415. DOI: 10.1007/978-3-540-72432-2_41. URL: https://doi.org/10.1007%2F978-3-540-72432-2_41.

[47] Broderick Crawford et al. "An Artificial Fish Swarm Optimization Algorithm to Solve Set Covering Problem". In: *Trends in Applied Knowledge-Based Systems and Data Science*. Springer International Publishing, 2016, pp. 892–903. DOI: 10.1007/978-3-319-42007-3_76. URL: https://doi.org/10.1007%2F978-3-319-42007-3_76.

[48] Broderick Crawford et al. "Application of the Artificial Bee Colony Algorithm for Solving the Set Covering Problem". In: *The Scientific World Journal* 2014 (2014), pp. 1–8. DOI: 10.1155/2014/189164. URL: https://doi.org/10.1155%2F2014%2F189164.

[49] Broderick Crawford et al. "Binary Firefly algorithm for the set covering problem". In: *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, June 2014. DOI: 10.1109/cisti.2014.6877090. URL: https://doi.org/10.1109%2Fcisti.2014.6877090.

[50] Broderick Crawford et al. "Fireworks Explosion Can Solve the Set Covering Problem". In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2016, pp. 477–490. DOI: 10.1007/978-3-319-33625-1_43. URL: https://doi.org/10.1007%2F978-3-319-33625-1_43.

[51] Broderick Crawford et al. "Solving the Set Covering Problem with a Shuffled Frog Leaping Algorithm". In: *Intelligent Information and Database Systems*. Springer International Publishing, 2015, pp. 41–50. DOI: 10.1007/978-3-319-15705-4_5. URL: https://doi.org/10.1007%2F978-3-319-15705-4_5.

[52] Mark S. Daskin and Edmund H. Stern. "A Hierarchical Objective Set Covering Model for Emergency Medical Service Vehicle Deployment". In: *Transportation Science* 15.2 (May 1981), pp. 137–152. DOI: 10.1287/trsc.15.2.137. URL: https://doi.org/10.1287%2Ftrsc.15.2.137.

[53] Pierre David, Vincent Idasiak, and Frédéric Kratz. "A Sensor Placement Approach for the Monitoring of Indoor Scenes". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 110–125. DOI: 10.1007/978-3-540-75696-5_7. URL: https://doi.org/10.1007%2F978-3-540-75696-5_7.

[54] E. W. Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1.1 (Dec. 1959), pp. 269–271. DOI: 10.1007/bf01386390. URL: https://doi.org/10.1007%2Fbf01386390.

[55] Xu Chu Ding, A.R. Rahmani, and M. Egerstedt. "Multi-UAV Convoy Protection: An Optimal Approach to Path Planning and Coordination". In: *IEEE Transactions on Robotics* 26.2 (Apr. 2010), pp. 256–268. DOI: 10.1109/tro.2010.2042325. URL: https://doi.org/10.1109%2Ftro.2010.2042325.

[56] Pinliang Dong. "Generating and updating multiplicatively weighted Voronoi diagrams for point, line and polygon features in GIS". In: *Computers & Geosciences* 34.4 (Apr. 2008), pp. 411–421. DOI: 10.1016/j.cageo.2007.04.005. URL: https://doi.org/10.1016%2Fj.cageo.2007.04.005.

[57] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents". In: *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* 26.1 (1996), pp. 29–41. DOI: 10.1109/3477.484436. URL: https://doi.org/10.1109%2F3477.484436.

[58] Martin Held. *Triangulation by Ear Clipping*. https://www.geometrictools.com/Documentation/TriangulationByEarClipping.pdf. Accessed: 2020-02-04.

[59] *earcut.hpp: Fast, header-only polygon triangulation*. https://github.com/mapbox/earcut.hpp. Accessed: 2020-02-04.

[60] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. "On the shape of a set of points in the plane". In: *IEEE Transactions on Information Theory* 29.4 (July 1983), pp. 551–559. DOI: 10.1109/tit.1983.1056714. URL: https://doi.org/10.1109%2Ftit.1983.1056714.

[61] Alon Efrat et al. "Sweeping Simple Polygons with a Chain of Guards". In: *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '00. San Francisco, California, USA: Society for Industrial and Applied Mathematics, 2000, pp. 927–936. ISBN: 0-89871-453-2. URL: http://dl.acm.org/citation.cfm?id=338219.338660.

[62] Ali O. Ercan et al. "Optimal Placement and Selection of Camera Network Nodes for Target Localization". In: *Distributed Computing in Sensor Systems*. Springer Berlin Heidelberg, 2006, pp. 389–404. DOI: 10.1007/11776178_24. URL: https://doi.org/10.1007%2F11776178_24.

[63] Uğur Murat Erdem and Stan Sclaroff. "Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements". In: *Computer Vision and Image Understanding* 103.3 (Sept. 2006), pp. 156–169. DOI: 10.1016/j.cviu.2006.06.005. URL: https://doi.org/10.1016%2Fj.cviu.2006.06.005.

[64] Anton V. Eremeev. "A Genetic Algorithm with a Non-Binary Representation for the Set Covering Problem". In: *Operations Research Proceedings 1998*. Springer Berlin Heidelberg, 1999, pp. 175–181. DOI: 10.1007/978-3-642-58409-1_17. URL: https://doi.org/10.1007%2F978-3-642-58409-1_17.

[65] Vladimir Estivill-Castro et al. "Illumination of polygons with vertex lights". In: *Information Processing Letters* 56.1 (Oct. 1995), pp. 9–13. DOI: 10.1016/0020-0190(95)00129-z. URL: https://doi.org/10.1016%2F0020-0190%2895%2900129-z.

[66] Javier Etcheberry. "The Set-Covering Problem: A New Implicit Enumeration Algorithm". In: *Operations Research* 25.5 (Oct. 1977), pp. 760–772. DOI: 10.1287/opre.25.5.760. URL: https://doi.org/10.1287%2Fopre.25.5.760.

[67] Muzaffar M. Eusuff and Kevin E. Lansey. "Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm". In: *Journal of Water Resources Planning and Management* 129.3 (May 2003), pp. 210–225. DOI: 10.1061/(asce)0733-9496(2003)129:3(210). URL: https://doi.org/10.1061%2F%28asce%290733-9496%282003%29129%3A3%28210%29.

[68] Eduardo Penha Castro Fantini and Luiz Chaimowicz. "Coverage in Arbitrary 3D Environments: The Art Gallery Problem in Shooter Games". In: (Universidade Presbiteriana Mackenzie, Oct. 16–18, 2013). São Paulo: IEEE, Oct. 2013. URL: http://www.sbgames.org/sbgames2013/proceedings/comp/17-full-paper.pdf.

[69] Duc Fehr, Loren Fiore, and Nikolaos Papanikolopoulos. "Issues and solutions in surveillance camera placement". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2009. DOI: 10.1109/iros.2009.5354252. URL: https://doi.org/10.1109%2Firos.2009.5354252.

[70] Uriel Feige. "A threshold of ln n for approximating set cover". In: *Journal of the ACM* 45.4 (July 1998), pp. 634–652. DOI: 10.1145/285055.285059. URL: https://doi.org/10.1145%2F285055.285059.

[71] Giovanni Felici et al. "A-priori upper bounds for the set covering problem". In: *Annals of Operations Research* 238.1-2 (Dec. 2015), pp. 229–241. DOI: 10.1007/s10479-015-2069-0. URL: https://doi.org/10.1007%2Fs10479-015-2069-0.

[72] Thomas A. Feo and Mauricio G.C. Resende. "A probabilistic heuristic for a computationally difficult set covering problem". In: *Operations Research Letters* 8.2 (Apr. 1989), pp. 67–71. DOI: 10.1016/0167-6377(89)90002-3. URL: https://doi.org/10.1016%2F0167-6377%2889%2990002-3.

[73] Marshall L. Fisher. "An Applications Oriented Guide to Lagrangian Relaxation". In: *Interfaces* 15.2 (Apr. 1985), pp. 10–21. DOI: 10.1287/inte.15.2.10. URL: https://doi.org/10.1287%2Finte.15.2.10.

[74] Steve Fisk. "A short proof of Chvátal's Watchman Theorem". In: *Journal of Combinatorial Theory, Series B* 24.3 (June 1978), p. 374. DOI: 10.1016/0095-8956(78)90059-x. URL: https://doi.org/10.1016%2F0095-8956%2878%2990059-x.

[75] Martin Held. *FIST: Fast Industrial-Strength Triangulation of Polygons*. http://www.cosy.sbg.ac.at/~held/projects/triang/triang.html. Accessed: 2020-02-04.

[76] M. Flint, M. Polycarpou, and E. Fernandez-Gaucherand. "Cooperative control for multiple autonomous UAV's searching for targets". In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* IEEE, 2003. DOI: 10.1109/cdc.2002.1184272. URL: https://doi.org/10.1109%2Fcdc.2002.1184272.

[77] Yi-Ge Fu, Jie Zhou, and Lei Deng. "Surveillance of a 2D Plane Area with 3D Deployed Cameras". In: *Sensors* 14.2 (Jan. 2014), pp. 1988–2011. DOI: 10.3390/s140201988. URL: https://doi.org/10.3390%2Fs140201988.

[78] Giordano Fusco and Himanshu Gupta. "Placement and Orientation of Rotating Directional Sensors". In: *2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. IEEE, June 2010. DOI: 10.1109/secon.2010.5508238. URL: https://doi.org/10.1109%2Fsecon.2010.5508238.

[79] Chao Gao et al. "An efficient local search heuristic with row weighting for the unicost set covering problem". In: *European Journal of Operational Research* 246.3 (Nov. 2015), pp. 750–761. DOI: 10.1016/j.ejor.2015.05.038. URL: https://doi.org/10.1016%2Fj.ejor.2015.05.038.

[80] Fang Gao et al. "Application of improved discrete particle swarm algorithm in partner selection of virtual enterprise". In: *International Journal of Computer Science and Network Security* 6.3 (2006), pp. 208–212.

[81] Francisco Javier Martinez Garcia and José A. Moreno Perez. "Jumping frogs optimization: a new swarm method for discrete optimization". In: *Documentos de Trabajo del DEIOC* 3 (2008).

[82] Anouck R. Girard, Adam S. Howell, and J. Karl Hedrick. "Border patrol and surveillance missions using multiple unmanned air vehicles". In: *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*. IEEE, 2004. DOI: 10.1109/cdc.2004.1428713. URL: https://doi.org/10.1109%2Fcdc.2004.1428713.

[83] Fred Glover. "Heuristics for integer programming using surrogate constraints". In: *Decision Sciences* 8.1 (Jan. 1977), pp. 156–166. DOI: 10.1111/j.1540-5915.1977.tb01074.x. URL: https://doi.org/10.1111%2Fj.1540-5915.1977.tb01074.x.

[84] Fred Glover. "Tabu Search—Part I". In: *ORSA Journal on Computing* 1.3 (Aug. 1989), pp. 190–206. DOI: 10.1287/ijoc.1.3.190. URL: https://doi.org/10.1287%2Fijoc.1.3.190.

[85] Fred Glover. "Tabu Search—Part II". In: *ORSA Journal on Computing* 2.1 (Feb. 1990), pp. 4–32. DOI: 10.1287/ijoc.2.1.4. URL: https://doi.org/10.1287%2Fijoc.2.1.4.

[86] Héctor González-Banos. "A randomized art-gallery algorithm for sensor placement". In: *Proceedings of the seventeenth annual symposium on Computational geometry - SCG '01*. ACM Press, 2001. DOI: 10.1145/378583.378674. URL: https://doi.org/10.1145%2F378583.378674.

[87] Jose-Joel Gonzalez-Barbosa et al. "Optimal camera placement for total coverage". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, May 2009. DOI: 10.1109/robot.2009.5152761. URL: https://doi.org/10.1109%2Frobot.2009.5152761.

[88] Khronos Group. *OpenGL - The Industry Standard for High Performance Graphics*. 1992.

[89] Antonin Guttman. "R-trees". In: *Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD '84*. ACM Press, 1984. DOI: 10.1145/602259.602266. URL: https://doi.org/10.1145%2F602259.602266.

[90] Peter Hart, Nils Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/tssc.1968.300136. URL: https://doi.org/10.1109%2Ftssc.1968.300136.

[91] Anton van den Hengel et al. "Automatic camera placement for large scale surveillance networks". In: *2009 Workshop on Applications of Computer Vision (WACV)*. IEEE, Dec. 2009. DOI: 10.1109/wacv.2009.5403076. URL: https://doi.org/10.1109%2Fwacv.2009.5403076.

[92] C. A. R. Hoare. "Algorithm 64: Quicksort". In: *Communications of the ACM* 4.7 (July 1961), p. 321. DOI: 10.1145/366622.366644. URL: https://doi.org/10.1145%2F366622.366644.

[93] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. 1992. ISBN: 0262581116. URL: https://www.amazon.com/Adaptation-Natural-Artificial-Systems-Introductory/dp/0262581116.

[94] Ross Honsberger. *Mathematical Gems II (Dolciani Mathematical Expositions, No. 2) (Pt. 2)*. Mathematical Assn of Amer, 1976. ISBN: 0883853027. URL: https://www.amazon.com/Mathematical-Gems-Dolciani-Expositions-No/dp/0883853027.

[95] Eva Horster and Rainer Lienhart. "Approximating Optimal Visual Sensor Placement". In: *2006 IEEE International Conference on Multimedia and Expo*. IEEE, July 2006. DOI: 10.1109/icme.2006.262766. URL: https://doi.org/10.1109%2Ficme.2006.262766.

[96] *Calcul scientifique, services numériques de l'Université de Strasbourg*. https://services-numeriques.unistra.fr/les-services-aux-usagers/hpc.html. Accessed: 2020-10-04.

[97] IBM®. *IBM CPLEX Optimiser*. 1988.

[98] S. Indu et al. "Optimal sensor placement for surveillance of large spaces". In: *2009 Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*. IEEE, Aug. 2009. DOI: 10.1109/icdsc.2009.5289398. URL: https://doi.org/10.1109%2Ficdsc.2009.5289398.

[99] *Irrlicht Engine, a free open source 3D engine*. http://irrlicht.sourceforge.net/. Accessed: 2020-08-04.

[100] Larry W. Jacobs and Michael J. Brusco. "Note: A local-search heuristic for large set-covering problems". In: *Naval Research Logistics* 42.7 (Oct. 1995), pp. 1129–1140. DOI: 10.1002/1520-6750(199510)42:7<1129::aid-nav3220420711>3.0.co;2-m. URL: https://doi.org/10.1002%2F1520-6750%28199510%2942%3A7%3C1129%3A%3Aaid-nav3220420711%3E3.0.co%3B2-m.

[101] Firdaus Janoos et al. "Sensor configuration for coverage optimization for surveillance applications". In: *Videometrics IX*. Ed. by Jean-Angelo Beraldin, Fabio Remondino, and Mark R. Shortis. SPIE, Jan. 2007. DOI: 10.1117/12.704062. URL: https://doi.org/10.1117%2F12.704062.

[102] Adrián Jaramillo et al. "Solving the Set Covering Problem with the Soccer League Competition Algorithm". In: *Trends in Applied Knowledge-Based Systems and Data Science*. Springer International Publishing, 2016, pp. 884–891. DOI: 10.1007/978-3-319-42007-3_75. URL: https://doi.org/10.1007%2F978-3-319-42007-3_75.

[103] Ayush Joshi, Jonathan E. Rowe, and Christine Zarges. "An Immune-Inspired Algorithm for the Set Cover Problem". In: *Parallel Problem Solving from Nature – PPSN XIII*. Springer International Publishing, 2014, pp. 243–251. DOI: 10.1007/978-3-319-10762-2_24. URL: https://doi.org/10.1007%2F978-3-319-10762-2_24.

[104] Dervis Karaboga. *An idea based on honey bee swarm for numerical optimization*. Tech. rep. Erciyes University, 2005.

[105] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. Springer US, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9. URL: https://doi.org/10.1007%2F978-1-4684-2001-2_9.

[106] Giorgos D. Kazazakis and Antonis Argyros. "Fast positioning of limited-visibility guards for the inspection of 2D workspaces". In: *IEEE/RSJ International Conference on Intelligent Robots and System*. IEEE, 2002. DOI: 10.1109/irds.2002.1041701. URL: https://doi.org/10.1109%2Firds.2002.1041701.

[107] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. IEEE, 1995. DOI: 10.1109/icnn.1995.488968. URL: https://doi.org/10.1109%2Ficnn.1995.488968.

[108] Mojtaba Ahmadieh Khanesar, Mohammad Teshnehlab, and Mahdi Aliyari Shoorehdeli. "A novel binary particle swarm optimization". In: *2007 Mediterranean Conference on Control & Automation*. IEEE, June 2007. DOI: 10.1109/med.2007.4433821. URL: https://doi.org/10.1109%2Fmed.2007.4433821.

[109] Jongrae Kim and J L Crassidis. "UAV path planning for maximum visibility of ground targets in an urban area". In: *2010 13th International Conference on Information Fusion*. IEEE, July 2010. DOI: 10.1109/icif.2010.5711852. URL: https://doi.org/10.1109%2Ficif.2010.5711852.

[110] Jongrae Kim and Yoonsoo Kim. "Moving ground target tracking in dense obstacle areas using UAVs". In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 8552–8557. DOI: 10.3182/20080706-5-kr-1001.01446. URL: https://doi.org/10.3182%2F20080706-5-kr-1001.01446.

[111] Kamyoung Kim, Alan T Murray, and Ningchuan Xiao. "A multiobjective evolutionary algorithm for surveillance sensor placement". In: *Environment and Planning B: Planning and Design* 35.5 (2008), pp. 935–948. DOI: 10.1068/b33139. URL: https://doi.org/10.1068%2Fb33139.

[112] Derek Kingston. "Road surveillance using a team of small UAVs". In: *Unmanned Systems Technology XI*. Ed. by Grant R. Gerhart, Douglas W. Gage, and Charles M. Shoemaker. SPIE, May 2009. DOI: 10.1117/12.818774. URL: https://doi.org/10.1117%2F12.818774.

[113] Derek Kingston, Randal W. Beard, and Ryan S. Holt. "Decentralized Perimeter Surveillance Using a Team of UAVs". In: *IEEE Transactions on Robotics* 24.6 (Dec. 2008), pp. 1394–1404. DOI: 10.1109/tro.2008.2007935. URL: https://doi.org/10.1109%2Ftro.2008.2007935.

[114] Scott Kirkpatrick, C.D. Gelatt, and Mario P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (May 1983), pp. 671–680. DOI: 10.1126/science.220.4598.671. URL: https://doi.org/10.1126%2Fscience.220.4598.671.

[115] Krishna Reddy Konda and Nicola Conci. "Optimal configuration of PTZ camera networks based on visual quality assessment and coverage maximization". In: *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*. IEEE, Oct. 2013. DOI: 10.1109/icdsc.2013.6778202. URL: https://doi.org/10.1109%2Ficdsc.2013.6778202.

[116] Krishna Reddy Konda and Nicola Conci. "Real-time reconfiguration of PTZ camera networks using motion field entropy and visual coverage". In: *Proceedings of the International Conference on Distributed Smart Cameras - ICDSC '14*. ACM Press, 2014. DOI: 10.1145/2659021.2659051. URL: https://doi.org/10.1145%2F2659021.2659051.

[117] Harry Kornilakis and Panagiotis Stamatopoulos. "Crew Pairing Optimization with Genetic Algorithms". In: *Methods and Applications of Artificial Intelligence*. Springer Berlin Heidelberg, 2002, pp. 109–120. DOI: 10.1007/3-540-46014-4_11. URL: https://doi.org/10.1007%2F3-540-46014-4_11.

[118] Julien Kritter et al. "On the computational cost of the set cover approach for the optimal camera placement problem and possible trade-offs for surveillance infrastructure design". In: *RAIRO Operations Research* (2020). Submitted.

[119] Julien Kritter et al. "On the optimal placement of cameras for surveillance and the underlying set cover problem". In: *Applied Soft Computing* 74 (Jan. 2019), pp. 133–153. DOI: 10.1016/j.asoc.2018.10.025.

[120] Julien Kritter et al. "On the real-world applicability of state-of-the-art algorithms for the optimal camera placement problem". In: *6th 2019 International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, Apr. 2019. DOI: 10.1109/CoDIT.2019.8820295.

[121] Julien Kritter et al. "On the use of human-assisted optimisation for the optimal camera placement problem and the surveillance of urban events". In: *7th 2020 International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, June 2020.

[122] *La vidéosurveillance, vidéoprotection sur la voie publique*. https://www.cnil.fr/fr/la-videosurveillance-videoprotection-sur-la-voie-publique. Accessed: 2020-15-03. Commission Nationale de l'Informatique et des Libertés (CNIL).

[123] Guanghui Lan, Gail W. DePuy, and Gary E. Whitehouse. "An effective and simple heuristic for the set covering problem". In: *European Journal of Operational Research* 176.3 (Feb. 2007), pp. 1387–1403. DOI: 10.1016/j.ejor.2005.09.028. URL: https://doi.org/10.1016%2Fj.ejor.2005.09.028.

[124] Lucas Lessing, Irina Dumitrescu, and Thomas Stützle. "A Comparison Between ACO Algorithms for the Set Covering Problem". In: *Ant Colony Optimization and Swarm Intelligence*. Springer Berlin Heidelberg, 2004, pp. 1–12. DOI: 10.1007/978-3-540-28646-2_1. URL: https://doi.org/10.1007%2F978-3-540-28646-2_1.

[125] Junbin Liu, Sridha Sridharan, and Clinton Fookes. "Recent Advances in Camera Planning for Large Area Surveillance". In: *ACM Computing Surveys* 49.1 (May 2016), pp. 1–37. DOI: 10.1145/2906148. URL: https://doi.org/10.1145%2F2906148.

[126] Junbin Liu et al. "On the Statistical Determination of Optimal Camera Configurations in Large Scale Surveillance Networks". In: *Computer Vision – ECCV 2012*. Springer Berlin Heidelberg, 2012, pp. 44–57. DOI: 10.1007/978-3-642-33718-5_4. URL: https://doi.org/10.1007%2F978-3-642-33718-5_4.

[127] Junbin Liu et al. "Optimal Camera Planning Under Versatile User Constraints in Multi-Camera Image Processing Systems". In: *IEEE Transactions on Image Processing* 23.1 (Jan. 2014), pp. 171–184. DOI: 10.1109/tip.2013.2287606. URL: https://doi.org/10.1109%2Ftip.2013.2287606.

[128] Luiz A. N. Lorena and L. de Souza Lopes. "Genetic algorithms applied to computationally difficult set covering problems". In: *Journal of the Operational Research Society* 48.4 (Apr. 1997), pp. 440–445. DOI: 10.1057/palgrave.jors.2600380. URL: https://doi.org/10.1057%2Fpalgrave.jors.2600380.

[129] Vittorio Maniezzo. "Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem". In: *INFORMS Journal on Computing* 11.4 (Nov. 1999), pp. 358–369. DOI: 10.1287/ijoc.11.4.358. URL: https://doi.org/10.1287%2Fijoc.11.4.358.

[130] André L. Maravilha, Jaime A. Ramírez, and Felipe Campelo. "Combinatorial Optimization with Differential Evolution: A Set-based Approach". In: *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*. GECCO Comp '14. ACM, 2014, pp. 69–70. ISBN: 978-1-4503-2881-4. DOI: 10.1145/2598394.2598463. URL: http://doi.acm.org/10.1145/2598394.2598463.

[131] Elena Marchiori and Adri Steenbeek. "An Evolutionary Algorithm for Large Scale Set Covering Problems with Application to Airline Crew Scheduling". In: *Real-World Applications of Evolutionary Computing*. Springer Berlin Heidelberg, 2000, pp. 370–384. DOI: 10.1007/3-540-45561-2_36. URL: https://doi.org/10.1007%2F3-540-45561-2_36.

[132] Elena Marchiori and Adri Steenbeek. *An Iterated Heuristic Algorithm for the Set Covering Problem*. 1998.

[133] MathWorks®. *MATLAB*. 1984.

[134] Aaron Mavrinac and Xiang Chen. "Modeling Coverage in Camera Networks: A Survey". In: *International Journal of Computer Vision* 101.1 (Nov. 2012), pp. 205–226. DOI: 10.1007/s11263-012-0587-7. URL: https://doi.org/10.1007%2Fs11263-012-0587-7.

[135] Seyedali Mirjalili and Andrew Lewis. "S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization". In: *Swarm and Evolutionary Computation* 9 (Apr. 2013), pp. 1–14. DOI: 10.1016/j.swevo.2012.09.002. URL: https://doi.org/10.1016%2Fj.swevo.2012.09.002.

[136] Anurag Mittal and Larry S. Davis. In: *International Journal of Computer Vision* 51.3 (2003), pp. 189–203. DOI: 10.1023/a:1021849801764. URL: https://doi.org/10.1023%2Fa%3A1021849801764.

[137] Anurag Mittal and Larry S. Davis. "A General Method for Sensor Planning in Multi-Sensor Systems: Extension to Random Occlusion". In: *International Journal of Computer Vision* 76.1 (July 2007), pp. 31–52. DOI: 10.1007/s11263-007-0057-9. URL: https://doi.org/10.1007%2Fs11263-007-0057-9.

[138] Anurag Mittal and Larry S. Davis. "Visibility Analysis and Sensor Planning in Dynamic Environments". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 175–189. DOI: 10.1007/978-3-540-24670-1_14. URL: https://doi.org/10.1007%2F978-3-540-24670-1_14.

[139] Tomas Möller and Ben Trumbore. "Fast, Minimum Storage Ray-Triangle Intersection". In: *Journal of Graphics Tools* 2.1 (Jan. 1997), pp. 21–28. DOI: 10.1080/10867651.1997.10487468.

[140] Yacine Morsly, Mohand Said Djouadi, and Nabil Aouf. "On the best interceptor placement for an optimally deployed visual sensor network". In: *2010 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, Oct. 2010. DOI: 10.1109/icsmc.2010.5642200. URL: https://doi.org/10.1109%2Ficsmc.2010.5642200.

[141] Yacine Morsly et al. "Particle Swarm Optimization Inspired Probability Algorithm for Optimal Camera Network Placement". In: *IEEE Sensors Journal* 12.5 (May 2012), pp. 1402–1412. DOI: 10.1109/jsen.2011.2170833. URL: https://doi.org/10.1109%2Fjsen.2011.2170833.

[142] M. H. Mulati and A. A. Constantino. "Ant-Line: A Line-Oriented ACO Algorithm for the Set Covering Problem". In: *2011 30th International Conference of the Chilean Computer Science Society*. IEEE, Nov. 2011. DOI: 10.1109/sccc.2011.34. URL: https://doi.org/10.1109%2Fsccc.2011.34.

[143] Vikram P. Munishwar and Nael B. Abu-Ghazaleh. "Coverage algorithms for visual sensor networks". In: *ACM Transactions on Sensor Networks* 9.4 (July 2013), pp. 1–36. DOI: 10.1145/2489253.2489262. URL: https://doi.org/10.1145%2F2489253.2489262.

[144] Vikram P. Munishwar and Nael B. Abu-Ghazaleh. "Scalable target coverage in smart camera networks". In: *Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras - ICDSC '10*. ACM Press, 2010. DOI: 10.1145/1865987.1866020. URL: https://doi.org/10.1145%2F1865987.1866020.

[145] Alan T. Murray et al. "Coverage optimization to support security monitoring". In: *Computers, Environment and Urban Systems* 31.2 (Mar. 2007), pp. 133–147. DOI: 10.1016/j.compenvurbsys.2006.06.002. URL: https://doi.org/10.1016%2Fj.compenvurbsys.2006.06.002.

[146] Nysret Musliu. "Local Search Algorithm for Unicost Set Covering Problem". In: *Advances in Applied Artificial Intelligence*. Springer Berlin Heidelberg, 2006, pp. 302–311. DOI: 10.1007/11779568_34. URL: https://doi.org/10.1007%2F11779568_34.

[147] Zahra Naji-Azimi, Paolo Toth, and Laura Galli. "An electromagnetism metaheuristic for the unicost set covering problem". In: *European Journal of Operational Research* 205.2 (Sept. 2010), pp. 290–300. DOI: 10.1016/j.ejor.2010.01.035. URL: https://doi.org/10.1016%2Fj.ejor.2010.01.035.

[148] Prabhu Natarajan et al. "Decision-theoretic coordination and control for active multi-camera surveillance in uncertain, partially observable environments". In: *2012 Sixth International Conference on Distributed Smart Cameras (ICDSC)*. Oct. 2012, pp. 1–6. ISBN: 978-1-4503-1772-6.

[149] Prabhu Natarajan et al. "Scalable Decision-Theoretic Coordination and Control for Real-time Active Multi-Camera Surveillance". In: *Proceedings of the International Conference on Distributed Smart Cameras - ICDSC '14*. ACM Press, 2014. DOI: 10.1145/2659021.2659042. URL: https://doi.org/10.1145%2F2659021.2659042.

[150] John A. Nelder and Roger Mead. "A Simplex Method for Function Minimization". In: *The Computer Journal* 7.4 (Jan. 1965), pp. 308–313. DOI: 10.1093/comjnl/7.4.308. URL: https://doi.org/10.1093%2Fcomjnl%2F7.4.308.

[151] Nikhil Nigam and Ilan Kroo. "Persistent Surveillance Using Multiple Unmanned Air Vehicles". In: *2008 IEEE Aerospace Conference*. IEEE, Mar. 2008. DOI: 10.1109/aero.2008.4526242. URL: https://doi.org/10.1109%2Faero.2008.4526242.

[152] Nikhil Nigam et al. “Control of Multiple UAVs for Persistent Surveillance: Algorithm and Flight Test Results”. In: *IEEE Transactions on Control Systems Technology* 20.5 (Sept. 2012), pp. 1236–1251. DOI: 10.1109/tcst.2011.2167331. URL: https://doi.org/10.1109%2Ftcst.2011.2167331.

[153] Joseph O’Rourke. *Art Gallery Theorems and Algorithms*. Vol. 3. International Series of Monographs on Computer Science. Oxford University Press, 1987. ISBN: 0195039653. URL: https://www.amazon.com/Theorems-Algorithms-International-Monographs-Computer/dp/0195039653.

[154] Gustavo Olague and Roger Mohr. “Optimal camera placement for accurate reconstruction”. In: *Pattern Recognition* 35.4 (Apr. 2002), pp. 927–944. DOI: 10.1016/s0031-3203(01)00076-0. URL: https://doi.org/10.1016%2Fs0031-3203%2801%2900076-0.

[155] *OpenStreetMap*. https://www.openstreetmap.org/. Accessed: 2020-31-03.

[156] *OSM2World*. http://osm2world.org/. Accessed: 2020-11-04.

[157] Abhay K. Parekh. “A note on the greedy approximation algorithm for the unweighted set covering problem”. In: (1988).

[158] Mark A. Peot et al. “Planning sensing actions for UAVs in urban domains”. In: *Unmanned/Unattended Sensors and Sensor Networks II*. Ed. by Edward M. Carapezza. SPIE, Oct. 2005. DOI: 10.1117/12.634899. URL: https://doi.org/10.1117%2F12.634899.

[159] Maxime Pinard et al. “A Memetic Approach for the Unicost Set Covering Problem”. In: *Proceedings of the 14th Learning And Intelligent Optimization Conference*. Athens, Greece, June 2020.

[160] Luigi Di Puglia Pugliese et al. “Modelling the mobile target covering problem using flying drones”. In: *Optimization Letters* 10.5 (Aug. 2015), pp. 1021–1052. DOI: 10.1007/s11590-015-0932-1. URL: https://doi.org/10.1007%2Fs11590-015-0932-1.

[161] Fahd Rafi et al. “Autonomous target following by unmanned aerial vehicles”. In: *Unmanned Systems Technology VIII*. Ed. by Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage. SPIE, May 2006. DOI: 10.1117/12.667356. URL: https://doi.org/10.1117%2F12.667356.

[162] Maher Rebai et al. “A Branch and Bound Algorithm for the Critical Grid Coverage Problem in Wireless Sensor Networks”. In: *International Journal of Distributed Sensor Networks* 10.2 (Jan. 2014), p. 769658. DOI: 10.1155/2014/769658. URL: https://doi.org/10.1155%2F2014%2F769658.

[163] Maher Rebai et al. “Exact Biobjective Optimization Methods for Camera Coverage Problem in Three-Dimensional Areas”. In: *IEEE Sensors Journal* 16.9 (May 2016), pp. 3323–3331. DOI: 10.1109/jsen.2016.2519451. URL: https://doi.org/10.1109%2Fjsen.2016.2519451.

[164] Maher Rebai et al. “Sensor deployment optimization methods to achieve both coverage and connectivity in wireless sensor networks”. In: *Computers & Operations Research* 59 (July 2015), pp. 11–21. DOI: 10.1016/j.cor.2014.11.002. URL: https://doi.org/10.1016%2Fj.cor.2014.11.002.

[165] Zhi-Gang Ren et al. “New ideas for applying ant colony optimization to the set covering problem”. In: *Computers & Industrial Engineering* 58.4 (May 2010), pp. 774–784. DOI: 10.1016/j.cie.2010.02.011. URL: https://doi.org/10.1016%2Fj.cie.2010.02.011.

[166] Victor Reyes et al. “A Beam-Search Approach to the Set Covering Problem”. In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2016, pp. 395–402. DOI: 10.1007/978-3-319-33625-1_35. URL: https://doi.org/10.1007%2F978-3-319-33625-1_35.

[167] Sumantra Dutta Roy, Santanu Chaudhury, and Subhashis Banerjee. “Active recognition through next view planning: a survey”. In: *Pattern Recognition* 37.3 (Mar. 2004), pp. 429–446. DOI: 10.1016/j.patcog.2003.01.002. URL: https://doi.org/10.1016%2Fj.patcog.2003.01.002.

[168] Juan Salas et al. “Binary Harmony Search Algorithm for Solving Set-Covering Problem”. In: *Trends in Applied Knowledge-Based Systems and Data Science*. Springer International Publishing, 2016, pp. 917–930. DOI: 10.1007/978-3-319-42007-3_78. URL: https://doi.org/10.1007%2F978-3-319-42007-3_78.

[169] Ville Satopaa et al. “Finding a ”Kneedle” in a Haystack: Detecting Knee Points in System Behavior”. In: *2011 31st International Conference on Distributed Computing Systems Workshops*. IEEE, June 2011. DOI: 10.1109/icdcsw.2011.20. URL: https://doi.org/10.1109%2Ficdcsw.2011.20.

[170] Ketan Savla, Francesco Bullo, and Emilio Frazzoli. “The coverage problem for loitering Dubins vehicles”. In: *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007. DOI: 10.1109/cdc.2007.4435017. URL: https://doi.org/10.1109%2Fcdc.2007.4435017.

[171] Mac Schwager, Brian J. Julian, and Daniela Rus. “Optimal coverage for multiple hovering robots with downward facing cameras”. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, May 2009. DOI: 10.1109/robot.2009.5152815. URL: https://doi.org/10.1109%2Frobot.2009.5152815.

[172] Sameh Al-Shihabi, Mazen Arafeh, and Mahmoud Barghash. “An improved hybrid algorithm for the set covering problem”. In: *Computers & Industrial Engineering* 85 (July 2015), pp. 328–334. DOI: 10.1016/j.cie.2015.04.007. URL: https://doi.org/10.1016%2Fj.cie.2015.04.007.

[173] A. Sinha, T. Kirubarajan, and Y. Bar-Shalom. “Autonomous surveillance by multiple cooperative UAVs”. In: *Signal and Data Processing of Small Targets 2005*. Ed. by Oliver E. Drummond. SPIE, Aug. 2005. DOI: 10.1117/12.619266. URL: https://doi.org/10.1117%2F12.619266.

[174] Mauricio Solar, Víctor Parada, and Rodrigo Urrutia. “A parallel genetic algorithm to solve the set-covering problem”. In: *Computers & Operations Research* 29.9 (Aug. 2002), pp. 1221–1235. DOI: 10.1016/s0305-0548(01)00026-0. URL: https://doi.org/10.1016%2Fs0305-0548%2801%2900026-0.

[175] Ricardo Soto et al. “Solving the non-unicost set covering problem by using cuckoo search and black hole optimization”. In: *Natural Computing* 16.2 (Jan. 2017), pp. 213–229. DOI: 10.1007/s11047-016-9609-7. URL: https://doi.org/10.1007%2Fs11047-016-9609-7.

[176] *Shuttle Radar Topography Mission*. https://www2.jpl.nasa.gov/srtm/. Accessed: 2020-31-03.

[177] Rainer Storn and Kenneth Price. In: *Journal of Global Optimization* 11.4 (1997), pp. 341–359. DOI: 10.1023/a:1008202821328. URL: https://doi.org/10.1023%2Fa%3A1008202821328.

[178] Thomas Stützle and Holger H. Hoos. "$\mathcal{MAX}$-$\mathcal{MIN}$ Ant System". In: *Future Generation Computer Systems* 16.8 (June 2000), pp. 889–914. DOI: 10.1016/s0167-739x(00)00043-1. URL: https://doi.org/10.1016%2Fs0167-739x%2800%2900043-1.

[179] Thomas Stützle and Holger H. Hoos. "The $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System and Local Search for Combinatorial Optimization Problems". In: *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Springer US, 1999, pp. 313–329. DOI: 10.1007/978-1-4615-5775-3_22. URL: https://doi.org/10.1007%2F978-1-4615-5775-3_22.

[180] Shyam Sundar and Alok Singh. "A hybrid heuristic for the set covering problem". In: *Operational Research* 12.3 (Sept. 2010), pp. 345–365. DOI: 10.1007/s12351-010-0086-y. URL: https://doi.org/10.1007%2Fs12351-010-0086-y.

[181] Ichiro Suzuki et al. "Searching a polyonal region from the boundary". In: *International Journal of Computational Geometry & Applications* 11.05 (Oct. 2001), pp. 529–553. DOI: 10.1142/s0218195901000638. URL: https://doi.org/10.1142%2Fs0218195901000638.

[182] Zhijun Tang and U. Ozguner. "Motion planning for multitarget surveillance with mobile sensor agents". In: *IEEE Transactions on Robotics* 21.5 (Oct. 2005), pp. 898–908. DOI: 10.1109/tro.2005.847567. URL: https://doi.org/10.1109%2Ftro.2005.847567.

[183] Konstantinos A. Tarabanis, Peter K. Allen, and Roger Y. Tsai. "A survey of sensor planning in computer vision". In: *IEEE Transactions on Robotics and Automation* 11.1 (1995), pp. 86–104. DOI: 10.1109/70.345940. URL: https://doi.org/10.1109%2F70.345940.

[184] Masruba Tasnim, Shahriar Rouf, and M. Sohel Rahman. "A CLONALG-based Approach for the Set Covering Problem". In: *Journal of Computers* 9.8 (Aug. 2014). DOI: 10.4304/jcp.9.8.1787-1795. URL: https://doi.org/10.4304%2Fjcp.9.8.1787-1795.

[185] Claudio Valenzuela et al. "A 2-level Metaheuristic for the Set Covering Problem". In: *International Journal of Computers Communications & Control* 7.2 (Sept. 2014), p. 377. DOI: 10.15837/ijccc.2012.2.1417. URL: https://doi.org/10.15837%2Fijccc.2012.2.1417.

[186] Francis J. Vasko, Yun Lu, and Kenneth Zyma. "What is the best greedy-like heuristic for the weighted set covering problem?" In: *Operations Research Letters* 44.3 (May 2016), pp. 366–369. DOI: 10.1016/j.orl.2016.03.007. URL: https://doi.org/10.1016%2Fj.orl.2016.03.007.

[187] Richard Wise and Rolf Rysdyk. "UAV Coordination for Autonomous Target Tracking". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug. 2006. DOI: 10.2514/6.2006-6453. URL: https://doi.org/10.2514%2F6.2006-6453.

[188] Yi-Chun Xu, Bangjun Lei, and Emile A. Hendriks. "Constrained particle swarm algorithms for optimizing coverage of large-scale camera networks with mobile nodes". In: *Soft Computing* 17.6 (Jan. 2013), pp. 1047–1057. DOI: 10.1007/s00500-012-0978-2. URL: https://doi.org/10.1007%2Fs00500-012-0978-2.

[189] Reda Yaagoubi et al. “HybVOR: A Voronoi-Based 3D GIS Approach for Camera Surveillance Network Placement”. In: *ISPRS International Journal of Geo-Information* 4.2 (May 2015), pp. 754–782. DOI: 10.3390/ijgi4020754. URL: https://doi.org/10.3390%2Fijgi4020754.

[190] Kenichi Yabuta and Hitoshi Kitazawa. “Optimum camera placement considering camera specification for security monitoring”. In: *2008 IEEE International Symposium on Circuits and Systems*. IEEE, May 2008. DOI: 10.1109/iscas.2008.4541867. URL: https://doi.org/10.1109%2Fiscas.2008.4541867.

[191] Mutsunori Yagiura, Masahiro Kishida, and Toshihide Ibaraki. “A 3-flip neighborhood local search for the set covering problem”. In: *European Journal of Operational Research* 172.2 (July 2006), pp. 472–499. DOI: 10.1016/j.ejor.2004.10.018. URL: https://doi.org/10.1016%2Fj.ejor.2004.10.018.

[192] Yi Yao et al. “Can You See Me Now? Sensor Positioning for Automated and Persistent Surveillance”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 40.1 (Feb. 2010), pp. 101–115. DOI: 10.1109/tsmcb.2009.2017507. URL: https://doi.org/10.1109%2Ftsmcb.2009.2017507.

[193] Yi Yao et al. “Sensor planning for automated and persistent object tracking with multiple cameras”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2008. DOI: 10.1109/cvpr.2008.4587515. URL: https://doi.org/10.1109%2Fcvpr.2008.4587515.

[194] James J. Q. Yu, Albert Y. S. Lam, and Victor O. K. Li. “Chemical reaction optimization for the set covering problem”. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, July 2014. DOI: 10.1109/cec.2014.6900233. URL: https://doi.org/10.1109%2Fcec.2014.6900233.

[195] Boyu Zhang et al. “A differential evolution approach for coverage optimization of visual sensor networks with parallel occlusion detection”. In: *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, July 2016. DOI: 10.1109/aim.2016.7576941. URL: https://doi.org/10.1109%2Faim.2016.7576941.

[196] Hongguang Zhang et al. “An optimized placement algorithm for collaborative information processing at a wireless camera network”. In: *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, July 2013. DOI: 10.1109/icme.2013.6607594. URL: https://doi.org/10.1109%2Ficme.2013.6607594.

[197] Xin-Yuan Zhang et al. “Kuhn–Munkres Parallel Genetic Algorithm for the Set Cover Problem and Its Application to Large-Scale Wireless Sensor Networks”. In: *IEEE Transactions on Evolutionary Computation* 20.5 (Oct. 2016), pp. 695–710. DOI: 10.1109/tevc.2015.2511142. URL: https://doi.org/10.1109%2Ftevc.2015.2511142.

[198] Jian Zhao and Sen-ching S. Cheung. “Multi-Camera Surveillance with Visual Tagging and Generic Camera Placement”. In: *2007 First ACM/IEEE International Conference on Distributed Smart Cameras*. IEEE, Sept. 2007. DOI: 10.1109/icdsc.2007.4357532. URL: https://doi.org/10.1109%2Ficdsc.2007.4357532.

[199] Jian Zhao and Sen-ching S. Cheung. “Optimal visual sensor planning”. In: *2009 IEEE International Symposium on Circuits and Systems*. IEEE, May 2009. DOI: 10.1109/iscas.2009.5117711. URL: https://doi.org/10.1109%2Fiscas.2009.5117711.

[200] Jian Zhao et al. “Approximate Techniques in Solving Optimal Camera Placement Problems”. In: *International Journal of Distributed Sensor Networks* 9.11 (Jan. 2013), p. 241913. DOI: 10.1155/2013/241913. URL: https://doi.org/10.1155%2F2013%2F241913.

[201] Dimitrios Zorbas et al. “Energy Efficient Mobile Target Tracking Using Flying Drones”. In: *Procedia Computer Science* 19 (2013), pp. 80–87. DOI: 10.1016/j.procs.2013.06.016. URL: https://doi.org/10.1016%2Fj.procs.2013.06.016.

[202] Dimitrios Zorbas et al. “Optimal drone placement and cost-efficient target coverage”. In: *Journal of Network and Computer Applications* 75 (Jan. 2016), pp. 16–31. DOI: 10.1016/j.jnca.2016.08.009. URL: https://doi.org/10.1016%2Fj.jnca.2016.08.009.

Glossary

camera candidate A pair of discrete components (position and orientation) obtained from sampling possible camera configurations in a given surveillance area. 10

facial recognition In computer vision, facial recognition is the process of identifying a person based on images of their face. 5

feature extraction In computer vision, feature extraction is the process of isolating selected components (features) from an image for further interpretation and processing. 5

field of view The combination of the horizontal and vertical aperture angles which delimit a camera's frustum. 5

floodlight illumination problem Variant of the art gallery problem in which guards are replaced by light sources. The key difference is that a floodlight has a limited field of action, given as an angle: it therefore provides coverage in one direction only. 5

foreshortening Visual effect which causes measurements to appear shorter on an image than they actually are. 5

frustum The area or volume visible by a camera in a given configuration. 6

full coverage An optimal camera placement problem instance is solved at full coverage when every ground sample it contains has to be covered by at least one camera candidate in the solution, regardless of its importance or location. 24

gait classification In computer vision, gait classification is the process of identifying the current gait (walking, running, crawling, ...) of a person based on an image. 5

ground sample Discrete component (position) obtained from sampling a given surveillance area. Refered to as target in Chapter IV. 10

guard In the art gallery problem, a guard is a selected point in space from which 360-degree coverage must be taken into account with unlimited range. 4

hand-off rate The amount or proportion of overlapping coverage provided by a camera infrastructure to ensure the proper hand-off of targets. 5

next best view In photogrammetry or related fields, the next best view problem is that of determining the best position from which the next image should be taken given those that are already available. 5

node An OpenStreetMap node is a point in the geographic coordinate system which represents an element of interest on the map, or serves as a component to build ways. 25

pan The horizontal angle at which a camera is set up. 6

photogrammetry Field of research which focuses on the extraction of measurements from photographs. Among other things, it tackles the issue of constructing 3D models of scenes and objects from images. 5

point-in-triangle test Algorithm which determines whether or not a point lies within a triangle. 10

relation An OpenStreetMap relation is an association of several nodes or ways which gives them semantic meaning as a collection. They are typically used to bring together several buildings of the same institution or delimit areas which fall under the same administrative authority. 25

sampling procedure Sampling is the process of reducing a continuous portion of space into a set of discrete points which represent it. 9

simple polygon In geometry, a polygon is said to be simple if none of its edges intersect. 4

tag An OpenStreetMap tag a key-value pair which provides metadata related to the node, way or relation it is attached to. 25

target localisation The problem of determining the location of selected targets in an environment using sensors, typically cameras. 5

tilt The vertical angle at which a camera is set up. 6

UAV Unmanned aerial vehicles or drones are remotely-controled robots typically used for adaptive surveillance and monitoring. 6

watchmen route problem Variant of the art gallery problem in which only one moving guard is available. The problem is then that of computing the optimal route the guard must follow to cover the entire gallery during their shift. 5

way An OpenStreetMap way is an ordered sequence of nodes which forms a line or polygon to represent roads, buildings, open areas and so on. 25

List of Figures

List of Tables

List of Algorithms