RAIRO-Oper. Res. 55 (2021) 2069–2091
https://doi.org/10.1051/ro/2021092

RAIRO Operations Research
www.rairo-ro.org

ON THE COMPUTATIONAL COST OF THE SET COVER APPROACH FOR THE OPTIMAL CAMERA PLACEMENT PROBLEM AND POSSIBLE SET-COVER-INSPIRED TRADE-OFFS FOR SURVEILLANCE INFRASTRUCTURE DESIGN

Julien Kritter*, Mathieu Brévilliers, Julien Lepagnot and Lhassane Idoumghar

Abstract. The $\mathcal{NP}$-hard minimum set cover problem (SCP) is a very typical model to use when attempting to formalise optimal camera placement (OCP) applications. In a generic form, the OCP problem relates to the positioning of individual cameras such that the overall network is able to cover a given area while meeting a set of application-specific requirements (image quality, redundancy, ...) and optimising an objective, typically minimum cost or maximum coverage. In this paper, we focus on an application called global or persistent surveillance: camera networks which ensure full coverage of a given area. As preliminary work, an instance generation pipeline is proposed to create OCP instances from real-world data and solve them using existing literature. The computational cost of both the instance generation process and the solving algorithms however highlights a need for more efficient methods for decision makers to use in real-world settings. In this paper, we therefore propose to review the suitability of the approach, and more specifically to question two key elements: the impact of sampling frequencies and the importance of rigid full-coverage constraints. The results allow us to quickly provide decision makers with an overview of available solutions and trade-offs.

Mathematics Subject Classification. 90-05, 90-06, 90-08, 90C27, 90-10.

Received December 9, 2019. Accepted June 17, 2021.

1. Introduction

Optimal camera placement (OCP) is one of the core elements to consider when designing camera-based surveillance infrastructure. With various applications, from crowd movement analysis to threat management, these systems are now ubiquitous. It has therefore become paramount that these be not only efficient, but also that they incur as low a cost as possible.

The OCP can be generically formulated as follows. Given various constraints, usually related to coverage or image quality, and an objective to optimise (typically, the cost), how can one determine the set of positions and orientations which best meets the requirements?

Keywords. Combinatorial optimization, optimal camera placement, set cover problem.

IRIMAS, Université de Haute-Alsace, 12 Rue des Frères Lumière, 68 093 Mulhouse, France.
*Corresponding author: julien.kritter@uha.fr

From a combinatorial standpoint, the OCP can be seen as structurally identical to a well-known $\mathcal{NP}$-hard problem: minimum set cover (SCP). In previous surveying work [25], it was noted that while both problems have received attention from researchers, the two bodies of literature hardly ever benefit from each other. In this paper, we establish preliminary results for the optimal camera placement problem in the real world when applied to the surveillance of urban areas, and propose a trade-off approach to help decision makers balance cost and camera network efficiency. This idea is supported by observations which suggest that both sampling and solving times could be significantly improved by allowing for small relaxations of the problem's sometimes rigid constraints.

The main motivation for this work is the possible integration of such an approach into user-assisted decision support systems: software which allows a user to guide the search for a solution by taking in regular input. In order for this to be usable, the solving algorithms must be able to run quickly so as not to hinder interactibility. Work along those lines may actually be found in [27], for which this paper is a preliminary study.

We begin in Section 2 with prior and related work and a more complete presentation of the OCP and the SCP. In Section 3 we introduce our instance generation pipeline and its inner-workings. In Section 4 we design and run a benchmark of existing OCP and SCP literature and report on our results. Given the observed computational costs, we then suggest a relaxation approach for the full coverage constraint in Section 5, using the max-cover problem as a tool. Finally, in Section 6 we validate our hypotheses using crowd simulations in continuous space and OCP solutions acquired with various sampling frequencies and coverage rates. We conclude by proposing a methodology which could enable decision makers to balance cost and coverage in a reasonable time when designing camera-based surveillance infrastructure.

2. The optimal camera placement problem

2.1. Context, related work and applications

The optimal camera placement problem can be found in several fields of research and has many academic, industrial and military applications. This section focuses on this aspect and briefly reviews OCP literature and applications. It is mostly based on prior surveying work done on the topic [25], to which the reader can refer for more information.

The fields of research related to computer vision are particularly active when it comes to camera placement. Applications which focus on scene reconstruction are one example, as the OCP can have serious consequences on the quality of the output models [12,36]. Work done on feature extraction can also be noted. It has contributed by providing sensor models and by integrating image constraints directly into the optimisation model. For more on this point, the reader is referred to [29] for a review.

Closer to the scope of this paper are applications in target tracking and area coverage. A key contribution from the former is the idea of backup coverage or hand-off rates which helps design resilient sensor networks, less sensitive to target movement or network damage [7,41]. Area coverage literature, on the other hand, focuses on more generic surveillance systems. Key references include [14,20] who formulated the problem as a binary integer program while integrating basic visibility constraints.

Surveillance applications combine several elements from the above but also bring components of their own. Subareas may be defined as more important, while others are less vital and can afford to go with fewer cameras and coverage accuracy. Image quality constraints can often be relaxed, since the network is usually set up to support interventions by law enforcement forces rather than serve as a standalone tool for activity and individual identification (which often raise more legal problems than they do technical ones).

2.2. Optimal camera placement for area coverage

The OCP for global coverage finds its roots in the popular Art Gallery Problem introduced by O'Rourke [39], although its specifics have significantly changed with time to include more realistic requirements and constraints. In this paper, we focus on one application of the OCP: global area surveillance. Our objective will therefore be

to determine camera locations and orientations which ensure complete coverage of the area while minimising the cost of the infrastructure (min-cost). Another approach to this problem is to maximise coverage with a fixed budget available for the network (max-cover).

Although this problem might appear suited for the continuous domain, very little work has actually been done along these lines. The discrete approach is more common, the idea being to *sample* the surveillance area into points, and camera configurations into so-called *candidates* each with a given set of position and orientation coordinates. The terms *sample* and *candidate* will therefore be used in this paper to represent the discrete components of the problem. A candidate can have several samples within range, and a sample can been seen by several candidates. Given these terms, the objective is to select the smallest subset of candidates which covers all the samples.

$$\min \sum_{j \in \mathcal{J}} c_j x_j \tag{2.1}$$

$$\sum_{j \in \mathcal{J}} a_{ij} x_j \geq 1 \qquad \forall i \in \mathcal{I} \tag{2.2}$$

$$x_j \qquad \in \mathcal{N} \qquad \forall j \in \mathcal{J}. \tag{2.3}$$

The typical formulation for this problem is given by equations (2.1) and (2.2), in which $\mathcal{I}$ is the set of all samples and $\mathcal{J}$ that of all candidates, each represented by a subset of $\mathcal{I}$. x_j is the binary decision variable for candidate $j \in \mathcal{J}$ while c_j represents its cost. a_{ij} is part of the input and is set to 1 if and only if candidate j covers or sees sample i.

The A matrix (a_{ij} elements) is usually pre-computed by testing all pairings in $\mathcal{I} \times \mathcal{J}$ in a process referred to as visibility analysis. Put simply, an algorithm is expected to run various checks in the environment and determine whether or not candidate j can cover sample i. These checks can range from simplistic 2D approaches which represent a camera's cover (or frustum) as a 2D triangle or parallelepiped [34, 44] to more advanced techniques which leverage the capability of full-blown 3D rendering tools and video game engines [2, 15].

In terms of modelling and solving, one of the first papers to consider complete area coverage while minimising network cost is credited to Horster and Lienhart [20], although Erdem and Sclaroff [14] provide an applicable and more general approach. The paper uses a simple Branch-and-Bound approach with very quickly highlights the need for more efficient algorithms. Nonetheless, the problem has been extensively studied, with numerous variants such as those modelled by Murray *et al.* [35], in which exactly p cameras must be distributed over regions of varying importance. This of course is a slight variation of the original combinatorial model, but a very common one in coverage applications. Moreover, the idea of partitioning the area into weighted regions (and defining so-called essential regions) is also very recurrent, Conci and Lizzi [11], Indu *et al.* [22] being two more examples.

Surveillance applications with a clearer focus on security have later emerged, 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 *et al.* [33] 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 [24] consider the possibility of network reconfigurations in scenarios involving the sudden shutting down of cameras. With similar concerns, Rebai *et al.* [41] 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 [45], and suggest the design of a collaborative network of nodes for information processing.

There are, of course, many more components of this problem which would be of interest for a review. Such a paper is actually part of our earlier work [25].

2.3. Relationship to the set cover problem

The formulation introduced in Section 2.2 will be familiar to those studying combinatorial optimisation: it is the standard binary integer program for the set covering problem. In graph theory, the problem is called dominating set. It is more generally formulated as follows:

Problem (Minimum set cover)**.** Given a set of elements $\mathcal{I}$ to be covered, and a collection of subsets $\mathcal{J}$ (columns) such that the union of all sets in $\mathcal{J}$ is $\mathcal{I}$, find a collection of columns $\mathcal{C}_1, \ldots, \mathcal{C}_k$ such that $\bigcup_{j=1}^{k} \mathcal{C}_j$ is $\mathcal{I}$. In other words, identify the smallest subset of $\mathcal{J}$ which covers $\mathcal{I}$. A common variant involves assigning a cost to each column and finding the subset which ensures minimum cost.

From the statement given in Section 2.2, the connection between the OCP and the SCP is rather straightforward and can be summarised by a description of the visibility analysis algorithms used in each study. Given the scale at which OCP instances can grow even from a simple change in sampling frequencies however, the process can end up taking a significant amount of time. This becomes even more of a problem when application-specific constraints are added (image quality requirements, privacy considerations, legal issues, and so on).

To tackle the curse of dimensionality when dealing with larges instances, Beasley [3] proposed two reduction procedures which allows instance generators to eliminate unnecessary columns, whether as a solving preprocessing stage or as an online process supporting instance generation itself. The first, the domination check, identifies columns which are only subsets of others. A note however: care must be taken when tackling the SCP with costs, as a seemingly dominated column could also prove significantly cheaper than its larger alternative. The second procedure, the inclusion check, identifies columns which have a monopoly over some rows. Given constraints (2.2), such columns are necessarily part of the solution and can therefore be removed from the instance before optimisation.

A review of the minimum set covering problem requires a paper of its own, so we again refer our reader to [25] for such work. The problem has generated a lot of research since the seminal works of Karp [23] and Chvátal [10]. Numerous deterministic and stochastic methods may be found, and Section 4 will actually provide a summary of the state of the art. Nonetheless, significant contributions include [3, 5, 6, 17, 18].

3. Camera placement in real-world urban areas

3.1. Motivations

Looking at the literature for our variant of the OCP, two observations can be made. The first is that every proposition tackles a different variant and usually tailors an algorithm for it, hence making numerical comparisons difficult and prone to overfitting biases. The second is that various simplifying assumptions typically need to be made to make the problem accessible. Many papers therefore use parallelepipedic, two-dimensional, surveillance areas and allow cameras to be placed anywhere. Occlusion is also often ignored and scalability is limited to that of toy problems. Several papers have of course managed to lift some of these constraints, such as those mentioned at the end of Section 2.2.

One of the goals of this instance generation pipeline is to reach a middle ground between the scalability of the basic approaches and the realism of those which were just mentioned. To this end, two main data sources need to be mentioned: OpenStreetMap (OSM), which we use to define and sample the surveillance area, and NASA's SRTM data which allows us to transform flat OSM maps into 3D elevated models.

3.2. Sampling the surveillance area

OSM data is structured around four core entities: *nodes*, *ways*, *relations* and *tags*. Nodes represent points in the geodetic coordinate system while ways connect them to create polygonal shapes which can be combined with relations and tagged to represent complex pathways, buildings, open areas, and so on. For a more thorough description of this representation scheme, see [37, 38].

The first step towards an OCP instance is to move into Cartesian space to simplify later computations. As we will be working on fairly localised areas (to the planet's scale), we propose to use a two-step conversion using a plane tangent to the Earth as our coordinate system (East-North-Up or ENU). The process first moves the data from geodetic to Earth-Centred, Earth-Fixed (ECEF) coordinates, and then to 1 m-based ENU using a reference point (ϕ_0, λ_0) arbitrarily set to the centre of the surveillance area's bounding box. The process is summarised in equations (3.1)–(3.5), where (ϕ, λ) are the geodetic coordinates of the point (node) being translated, a and b are the Earth's major and minor ellipsoidal axes, n its curvature radius at latitude ϕ and $E = \left(\frac{b}{a}\right)^2$ its squared eccentricity.

The second step involves using NASA's elevation data, as we have used a flat map for reference and have now moved into a 2D "plane" (z being near zero). The SRTM data provides us with an elevation value in meters for every 30×30 m square on Earth. For more precision, intermediary values are computed using a gradient between a node's SRTM cell and its neighbours.

For every ground OSM entity, the sampling process can then begin. Line-based entities such as roads are sampled by hopping from node to node, dropping samples on the segments at a given *ground* frequency (f_g^l) and across the width of the road (frequency f_g^w). Polygon-based entities such as pedestrian areas require more effort and are first split into triangles. These can then be sampled by drawing parallel lines which connect the two longest edges at regular f_g^w intervals and by sampling those like roads with no width. Note that we are working in three dimensions, which means several samples are actually created whenever we stop along a line, at various altitudes: the bounds are named $a_g^{\min}$ and $a_g^{\max}$ and the associated frequency is f_g^a.

$$n = \frac{a^2}{\sqrt{a^2 \cos^2(\phi) + b^2 \sin^2(\phi)}} \tag{3.1}$$

$$ecef_{\phi\lambda} = \begin{pmatrix} n\cos(\phi)\cos(\lambda) \\ n\cos(\phi)\sin(\lambda) \\ nE\sin(\phi) \end{pmatrix} \tag{3.2}$$

$$d = (d_x, d_y, d_z) = ecef_{\phi\lambda} - ecef_{\phi_0\lambda_0} \tag{3.3}$$

$$t = d_x \cos(\lambda_0) + d_y \sin(\lambda_0) \tag{3.4}$$

$$enu_{\phi\lambda} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -d_x \sin(\lambda_0) + d_y \cos(\lambda_0) \\ -t\sin(\phi_0) + d_z \cos(\phi_0) \\ t\cos(\phi_0) + d_z \sin(\phi_0) \end{pmatrix}. \tag{3.5}$$

3.3. Identifying camera candidates

Now that set $\mathcal{I}$ has been defined, a first step can be taken towards $\mathcal{J}$: camera candidates can be sampled. For every structural OSM entity, possible locations are determined by hopping along walls and other *structures* with frequencies and bounds f_s^l, f_s^a, $a_s^{\min}$ and $a_s^{\max}$ analogous to those used in Section 3.2. For orientation and to avoid candidates pointing inwards or placed on a shared wall between buildings, the entire process actually iterates over triplets of nodes and computes angles with all adjacent walls as depicted in Figure 1. Candidates are then generated at the current node towards the previous node (angle α_p), at the current node towards the next wall (α_n) and along the next wall, this time with a 180° opening. Horizontal orientation (pan) is sampled at frequency f_s^p and vertical orientation (tilt) at frequency f_s^t, both typically expressed as fractions on π.

Figure 2 illustrates the results of the sampling process for both the ground samples and the candidates. The red prisms on and above the ground represent the samples while the blue pyramids attached to the walls are the candidates pointing in the direction of their individual orientation vectors.

3.4. Visibility analysis and occlusion

Visibility analysis is the process during which samples and candidates are matched, effectively creating the subsets in $\mathcal{J}$, one for each candidate. The approach we propose relies on three consecutive checks for every

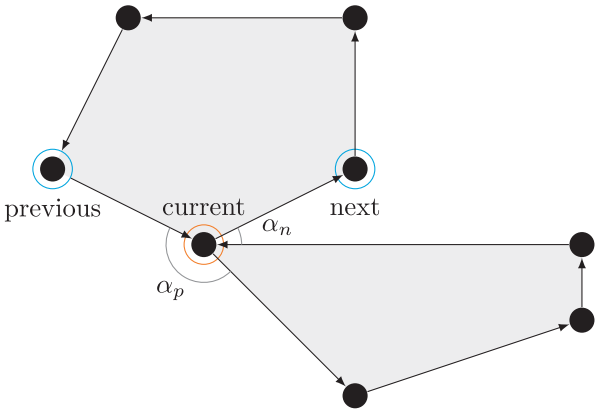

FIGURE 1. Orienting candidates during the sampling process.

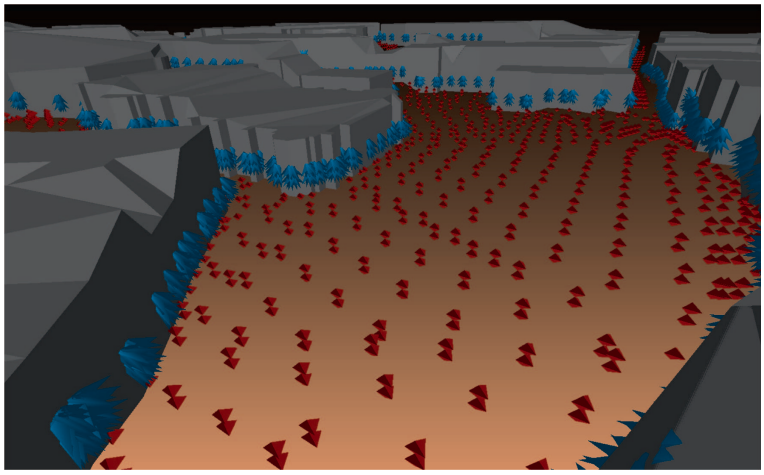

FIGURE 2. Result of the sampling procedure in MeshLab [31] for a subarea of instance `u2` (see Tab. 1).

sample-candidate pair: range, frustum and occlusion. Each test can end the process by setting $a_{ij} = 0$ (j cannot see i) while passing all three tests yields $a_{ij} = 1$ (j can see i).

The range check is straightforward: considering r, the camera model's range, sample i is considered within candidate j's range if and only if the two are r meters apart or less (their positions being known from the sampling process described in Sects. 3.2 and 3.3). Equation (3.6) defines r as a function of the camera's horizontal field of view F_h and resolution R_h, as well as a pixel-per-meter image quality requirement P.

$$r = \frac{R_h}{2P \cdot \tan\left(\frac{F_h}{2}\right)}. \tag{3.6}$$

Note that the first check does not take orientation into account. This is done by the second check, which computes the coordinates of the planes which delimit the candidate's frustum. This is achieved by first computing the plane which cuts the frustum in half vertically and rotating it around the vertical axis and the orientation vector to yield the sides of a 3D pyramid attached to the lens. All planes are represented in point-vector form with outward-facing normal vectors, meaning sample i must be at a "negative" distance of all of them to be visible.

Note that pyramid-shaped frustums are rather common in the literature [25] although the computation method may vary.

Finally, we propose the following approach to occlusion. We first use the OSM data to create a triangular mesh of the area which contains all possible sources of occlusion (buildings, walls, trees and so on). This is done using OSM2World [40] which we adjusted to use our ENU coordinate system. We then elevate the model using SRTM data and store all the vertices and faces which define it. Detecting occlusion between sample i and candidate j then becomes simple: if any triangle of the mesh intersects the vector between the two, the check fails. Note that special care must be taken for implementation to avoid checking faces that are clearly out of range. For the actual intersection test, we used the the popular Möller-Trumbore ray-triangle intersection algorithm [32].

4. Benchmarking existing algorithms

4.1. Parameters and instances

Using our approach, we generated 32 instances using parts of 8 European cities, listed in Table 2 along with the symbols we will be using later on and their OSM IDs in the OpenStreetMap database. Tables 3 and 4 give the sampling configurations (in meters and radians) which we used to generate these instances, such that, for example, instance `u2` refers to Mulhouse in configuration 2. Note that we used an image quality requirement of $P = 25$ pixels-per-meter, a camera resolution of 1920×1080 pixels and field-of-view angles of $65°$ in both directions. Table 1 summarised the key statistics of our instance set.

4.2. Algorithm selection and results

We propose to compare the following approaches on our instances. First, in order to define a baseline, we ran several basic algorithms: CPLEX's Branch-and-Cut [14, 20, 21], a row-oriented random algorithm and GRASP [17], a greedy algorithm which can be made less deterministic through its parameter $\alpha \in [0, 1]$ (from absolute randomness to absolute greediness). We also ran CPLEX's default simplex algorithm on the linear relaxation of our problem in order to obtain lower bounds.

Regarding the state of the art and based on the surveying work done in [26], we propose to include the following propositions. First, Marchiori and Steenbeek's ITEG algorithm as referenced by Andersen and Tirthapura [1] for their work on 3D sensor placement. For its efficiency on the OR-Library CLR instances [4], Meta-RaPS by Lan *et al.* [28]. For the STS instances, 3-FNLS by Yagiura *et al.* [43], the code for which was generously provided by the authors. As the overall best to date, RWLS by Gao *et al.* [18]. Finally, the OCP DEsim algorithm by Brévilliers *et al.* [9] for its efficiency on toy OCP instances. 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 3 reports on the average results obtained by every algorithm on our instance set.

4.3. Discussion

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 is not surprising for algorithms of this nature. The reader may refer to Table 5 for a summary of our initial lower and upper bounds.

Regarding GRASP, the algorithm shows poor performance when compared to the rest of the benchmark. Nevertheless, it provides a tighter upper bound on the problem, which might explain why so many propositions use such greedy algorithms 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 [10], such a conclusion cannot be drawn using our instances: the three sets of results are extremely similar (see Tab. 6). Looking at both best and average results, the pure greedy variant slightly

TABLE 1. Statistics for our first set of 32 real-world instances.

Symbol	Conf.	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{J}\rvert$	Density
b	0	9645	3854	0.0018
	1	12 526	4301	0.0018
	2	168 773	51 245	0.0010
	3	288 968	87 287	0.0010
k	0	8610	3829	0.0014
	1	12 374	4548	0.0013
	2	158 528	53 003	0.0010
	3	263 767	87 843	0.0009
l	0	13 881	5800	0.0012
	1	19 940	7608	0.0011
	2	190 677	61 722	0.0008
	3	309 447	98 309	0.0008
m	0	6862	2846	0.0028
	1	7978	2861	0.0034
	2	112 457	33 517	0.0017
	3	193 275	54 800	0.0016
n	0	2678	1160	0.0059
	1	2778	967	0.0087
	2	42 209	12 458	0.0044
	3	69 902	19 625	0.0040
u	0	900	446	0.0131
	1	1688	675	0.0100
	2	15 926	6255	0.0073
	3	28 927	10 898	0.0072
v	0	7840	3132	0.0019
	1	11 337	4054	0.0017
	2	214 874	47 462	0.0006
	3	374v447	79 237	0.0005
z	0	5027	1618	0.0042
	1	6218	1752	0.0047
	2	95 283	20 952	0.0022
	3	155 398	32 992	0.0022

TABLE 2. Locations used for instance generation.

Letter	Location (postal code)
b	Berlin Mitte (10117)
k	Kaiserslautern (67655)
l	City of London (various)
m	Mnchen city centre (80335)
r	Rennes city centre (various)
u	Mulhouse city centre (various)
v	Valbonne (06560)
z	Mainz city centre (55118)

TABLE 3. Ground sampling parameter values.

Cfg.	f_g^l	f_g^w	f_g^h	$a_g^{\min}$	$a_g^{\max}$
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 4. First set of candidate sampling parameter values.

Cfg.	f_s^l	f_s^h	$a_s^{\min}$	$a_s^{\max}$	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	$\frac{\pi}{6}$	$\frac{\pi}{6}$

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 3, 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 gain in performance is clearer and highlights the performance of CPLEX, RWLS and DEsim (see Tab. 7). 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 achieve such results for instances `u0` and `u1`. DEsim managed to stay reasonably close behind and joined the rankings for one more instance. Instances such as `u2`, `u3` and `n2` also isolate ITEG and Meta-RaPS in another cluster, however these algorithms still report poorer performance at a larger scale (see Tab. 8).

Some of these results may seem surprising for a problem we might have assumed to be, to some extent, harder to solve. 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 [9, 14, 20]. For 25 out of 32 instances however, CPLEX ranked first and reached optimality on 16. More importantly, it managed to find feasible solutions to all instances, something it could not always achieve on the OR-Library instances [4] or the toy instances used in [8, 9]. 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. This can be rather easily understood when looking at the generation process: 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 (Binary Integer Program) formulation, a solver only has a very limited set of variables to pick from.

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 (all costs set to 1) 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

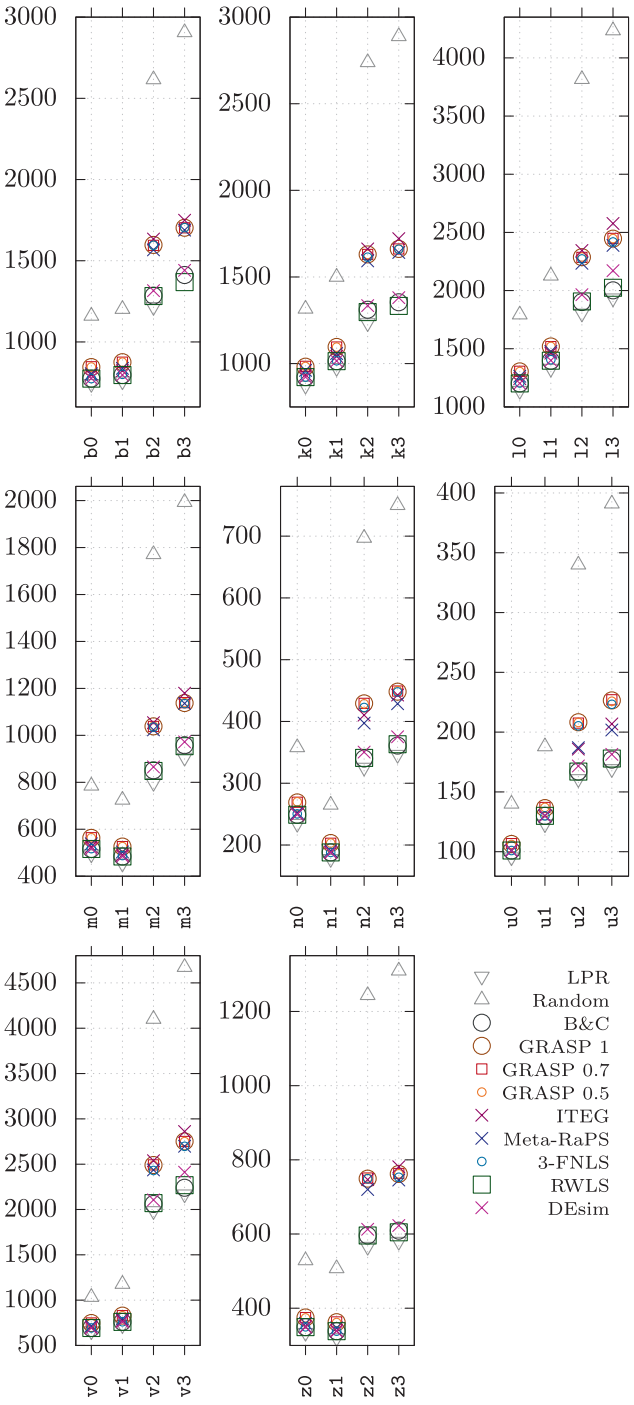

FIGURE 3. Average results (number of cameras) for every instance.

TABLE 5. Lower and upper bounds for our first set of instances.

Symbol	Conf.	LPR	Random (avg)	Rel. diff
b	0	742.19	1178.40	0.37
	1	758.45	1227.60	0.38
	2	1215.93	2686.90	0.55
	3	2.06	2969.10	1.00
k	0	873.20	1356.50	0.36
	1	975.57	1512.60	0.36
	2	1233.97	2792.90	0.56
	3	0.04	2940.10	1.00
l	0	1139.78	1829.80	0.38
	1	1329.45	2152.00	0.38
	2	1804.19	3910.30	0.54
	3	1934.51	4284.70	0.55
m	0	493.83	803.50	0.39
	1	454.58	742.90	0.39
	2	798.12	1806.90	0.56
	3	905.57	2045.20	0.56
n	0	234.59	372.50	0.37
	1	177.69	281.20	0.37
	2	325.43	720.60	0.55
	3	346.25	775.40	0.55
u	0	95.44	145.80	0.35
	1	123.18	201.90	0.39
	2	160.85	363.00	0.56
	3	169.23	408.50	0.59
v	0	658.57	1026.20	0.36
	1	726.45	1173.60	0.38
	2	1984.11	4097.70	0.52
	3	2168.83	4660.20	0.53
z	0	334.03	528.20	0.37
	1	319.63	507.40	0.37
	2	565.97	1252.20	0.55
	3	579.59	1312.20	0.56

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 an average duality gap of 3.1% (between 0.7% and 8%) and suggests that good and optimal solutions may often be surrounded by many alternatives with similar costs, making the gap more difficult to close.

5. Relaxing the full coverage constraint

5.1. The max-cover problem and its best $\mathcal{P}$-approximation

In Section 4, the benchmarked algorithms were all given one hour to complete each one of their runs. While in many cases, that budget turned out to be unnecessary, the hardest instances still pushed the algorithms to their limits: by the end of each run, small improvements were still being made on the solutions. However, as

TABLE 6. GRASP on our first set of instances (average results). The σ column is the standard deviation between all three configurations and highlights the low impact of the α parameter.

Symbol	Conf.	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 1$	σ
b	0	843.40	847.00	843.00	1.80
	1	874.60	874.10	873.30	0.54
	2	1594.20	1593.70	1598.70	2.25
	3	1704.40	1702.50	1702.40	0.92
k	0	984.30	982.70	982.30	0.86
	1	1093.20	1093.70	1094.20	0.41
	2	1630.30	1626.80	1631.10	1.87
	3	1658.30	1661.20	1663.60	2.17
l	0	1302.90	1305.50	1306.90	1.66
	1	1519.90	1518.00	1518.50	0.80
	2	2287.30	2282.40	2286.70	2.18
	3	2447.90	2442.70	2448.60	2.63
m	0	563.50	563.70	563.10	0.25
	1	526.40	524.40	525.30	0.82
	2	1038.20	1035.70	1037.50	1.05
	3	1139.50	1136.10	1139.90	1.70
n	0	271.10	269.80	270.10	0.56
	1	202.60	202.30	202.50	0.12
	2	429.40	429.50	430.30	0.40
	3	449.00	449.00	449.70	0.33
u	0	106.10	106.80	106.50	0.29
	1	136.50	137.40	137.00	0.37
	2	208.40	207.40	208.80	0.59
	3	225.70	225.80	227.00	0.59
v	0	745.90	748.30	746.70	1.00
	1	829.10	826.70	830.10	1.43
	2	2487.00	2493.80	2493.20	3.07
	3	2753.80	2750.60	2751.50	1.35
z	0	373.40	374.60	375.70	0.94
	1	362.10	363.30	362.00	0.59
	2	750.20	750.80	750.60	0.25
	3	761.10	764.90	765.10	1.84

discussed in Section 4.3, most of the solving time is spent on these minor improvements and algorithms usually manage to find reasonable solutions much quicker.

For this part of our work, we have chosen to regenerate our instances with more costly candidate sampling parameters (see Tab. 9), in order to explore harder instances. The new orientation parameters indeed enable the cameras to use significantly more of their range, which offers many more options for ground sample coverage. We have also chosen to remove the elevation data to better transition towards the simulation work done in Section 6. The Valbonne instances were also removed, since the surveillance area includes a lot of open spaces (forests) which cannot be monitored and therefore bring coverage rates to a point where the simulation data is hardly meaningful. Finally, since we are now allowing for partial coverage, inclusion checks (see Sect. 2.3) had to be disabled. This, along with improvements made on the instance generation codebase since [26], explain why differences may be observed in instance statistics between Table 1 in Section 4.1 and Table 10 in this section.

In this section, we consider the use of another combinatorial problem to find a good compromise between cost and coverage once some of the SCP cover constraints (Eq. (2.2)) are removed. The max-cover (or max-k-cover,

TABLE 7. DEsim, RWLS and CPLEX on our first set of instances (average results).

Symbol	Conf.	DEsim	RWLS	CPLEX
b	0	775.40	774.00	774.00
	1	796.70	796.00	796.00
	2	1316.10	1284.10	1285.00
	3	1440.70	1369.10	1412.00
k	0	922.10	922.00	922.00
	1	1015.00	1014.00	1014.00
	2	1334.60	1296.90	1311.00
	3	1381.40	1331.60	1354.00
l	0	1202.30	1202.00	1202.00
	1	1399.60	1397.00	1397.00
	2	1964.20	1904.70	1901.00
	3	2171.10	2023.90	2002.00
m	0	518.00	516.00	516.00
	1	485.00	484.00	484.00
	2	866.10	848.10	851.00
	3	972.90	953.40	958.00
n	0	249.80	249.00	249.00
	1	188.00	188.00	188.00
	2	350.60	341.10	340.00
	3	375.00	362.80	361.00
u	0	101.00	101.00	101.00
	1	130.00	130.00	130.00
	2	171.50	167.00	167.00
	3	181.60	177.90	177.00
v	0	693.00	693.00	693.00
	1	760.70	760.00	760.00
	2	2106.40	2069.50	2061.00
	3	2398.50	2269.60	2240.00
z	0	349.10	349.00	349.00
	1	338.00	338.00	338.00
	2	612.90	596.00	596.00
	3	624.20	604.40	609.00

MCP) problem is very similar, however this time the number k of columns (cameras) in the solution is set prior to solving. Given this limited budget, the objective becomes that of maximising coverage.

Problem (maximum k-cover)**.** Given an integer k, a set of elements $\mathcal{I}$ (rows), a collection of subsets $\mathcal{J}$ (columns) such that the union of all sets in $\mathcal{J}$ is $\mathcal{I}$, find a collection of columns $\mathcal{C}_1, \ldots, \mathcal{C}_k$ such that $|\bigcup_{j=1}^{k} \mathcal{C}_j|$ is maximal. In other words, identify a k-subset of $\mathcal{J}$ which covers the most elements from $\mathcal{I}$. A common variant involves assigning a positive weight to each row and finding the subset which ensures maximum weight.

While the search for a somewhat universally efficient algorithm for the SCP is still ongoing, research on the MCP now regards the greedy algorithm (Algorithm 1) as the best possible polynomial-time approximation of the problem [16], with an approximation ratio of $1 - \frac{1}{e}$. Therefore, the problem appears to be a good candidate as a tool to navigate trade-off solutions for the OCP. As a quick preview of the algorithm's performance, Table 11 reports on its coverage rates when k is set to the best-known full-coverage (SCP) solution (Tab. 12). In the worst-case (**instance k5**), the algorithm still covers 98.3% of the samples and terminates under 30 s on 2.5–2.67 GHz Intel processors.

Algorithm 1. The greedy algorithm for the maximum cover problem.

procedure MCP-Greedy($\mathcal{I}, \mathcal{J}, k$)
  $z \leftarrow$ an empty solution
  $\mathcal{I}_z = \emptyset$ (set of currently covered rows)
  $\mathcal{J}_z = J$ (set of eligible columns)
  $i \leftarrow 0$
  while $i < k$ **and** $|\mathcal{J}_z| > 0$ **do**
    $\mathcal{C}_p \leftarrow \arg\max_{\mathcal{C}_j \in \mathcal{J}_z} |\mathcal{C}_j \setminus \mathcal{I}_z|$ (column with the most uncovered elements)
    $z \leftarrow z \cup \{C_p\}$
    $\mathcal{I}_z \leftarrow \mathcal{I}_z \cup \mathcal{C}_p$
    $\mathcal{J}_z \leftarrow \mathcal{J}_z \setminus \{\mathcal{C}_p\}$
    $i \leftarrow i + 1$
  end while
  return z
end procedure

Table 8. Meta-RaPS, ITEG and 3-FNLS on our first set of instances (average results).

Symbol	Conf.	Meta-RaPS	ITEG	3-FNLS
b	0	805.20	793.30	774.20
	1	837.80	821.60	800.30
	2	1564.70	1632.80	1595.00
	3	1686.70	1753.30	1704.00
k	0	961.20	936.40	922.00
	1	1057.30	1039.80	1015.70
	2	1593.20	1663.10	1613.90
	3	1641.30	1724.70	1660.00
l	0	1263.70	1243.40	1204.90
	1	1476.20	1467.70	1405.80
	2	2230.50	2337.80	2272.90
	3	2392.10	2574.60	2416.90
m	0	538.80	525.90	517.50
	1	503.60	491.60	489.00
	2	1022.70	1055.50	1035.00
	3	1138.70	1178.70	1135.00
n	0	252.40	249.80	250.30
	1	190.40	188.00	188.00
	2	397.50	411.00	421.10
	3	429.10	440.80	448.00
u	0	101.00	101.00	101.00
	1	130.00	130.00	130.00
	2	187.00	186.10	202.90
	3	201.70	209.80	222.50
v	0	717.60	700.50	693.80
	1	794.20	775.30	761.60
	2	2436.90	2537.60	2430.80
	3	2693.40	2844.40	2697.80
z	0	359.20	350.30	349.80
	1	347.00	339.30	338.10
	2	721.40	747.80	749.00
	3	746.90	782.30	753.00

TABLE 9. Second set of candidate sampling parameter values.

Cfg.	f_s^l	f_s^h	$a_s^{\min}$	$a_s^{\max}$	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 10. Statistics for our second set of 28 real-world instances.

Symbol	Conf.	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{J}\rvert$	Density	Generation time
b	4	45 797	14 288	0.0011	00h 11m 05.79s
	5	66 913	25 562	0.0011	00h 25m 56.08s
	6	269 158	51 836	0.0010	08h 25m 50.67s
	7	460 703	79 899	0.0010	17h 55m 59.42s
k	4	42 679	13 114	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	60 230	17 002	0.0008	00h 05m 58.58s
	5	87 792	32 241	0.0008	00h 21m 13.65s
	6	334 519	74 494	0.0007	06h 02m 36.11s
	7	588 612	109 137	0.0007	08h 41m 47.76s
m	4	33 169	8915	0.0017	00h 03m 14.80s
	5	48 285	16 206	0.0017	00h 08m 28.35s
	6	191 366	38 927	0.0015	03h 36m 47.60s
	7	329 800	55 871	0.0015	07h 09m 46.58s
n	4	12 561	3436	0.0044	00h 02m 01.06s
	5	18 242	6244	0.0044	00h 02m 51.24s
	6	70 279	13 958	0.0042	01h 03m 50.87s
	7	120 786	19 430	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	25 700	6650	0.0070	00h 27m 21.06s
	7	45 691	9458	0.0069	01h 23m 12.48s
z	4	27 390	6102	0.0021	00h 02m 02.61s
	5	39 410	11 286	0.0022	00h 02m 40.98s
	6	156 439	22 550	0.0022	01h 10m 26.82s
	7	268 640	32 692	0.0021	01h 57m 32.43s

5.2. Solutions front and curve knees

A first step towards understanding the layout of the solutions available between full coverage and single camera is to determine a way to visualise the solutions front: a set of solutions spanning across the spectrum and which drop a varying number of constraints (samples). The approach used in this work is as follows.

First, the extrema of the front are computed. The full coverage point is a solution which covers all samples in $\mathcal{I}$ with the minimum number of cameras. In order to get a slightly broader front and to locate the optimal SCP solution on it, we choose to use the greedy SCP solution here (see Algorithm 2). The single candidate point is rather trivial: it is the solution to the max-1-cover instance, which therefore solely contains the candidate in $\mathcal{J}$ which covers the most points.

TABLE 11. MCP greedy coverage rates when setting k to the SCP optimal value.

	City						
Cfg.	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 12. Best-known full-coverage (SCP) solutions to the real-world instances.

	City						
Cfg.	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

Once these solutions are found, the front can be computed in the following way. 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 a max-k-cover solution with $k = 1 + f$. The other points are defined incrementally up to $k = |z| - f$ and computed using Algorithm 2.

Figure 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 coverage rate. The vertical lines are the best known full coverage solutions reported in Table 12. The squares are the curve knees, defined as the points of strongest deviation and computed using the Kneedle algorithm designed by Satopaa *et al.* [42]. 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 4 shows, this section spans across the majority of the cost range.

Algorithm 2. The greedy algorithm for the minimum set cover problem.

procedure SCP-GREEDY($\mathcal{I}, \mathcal{J}$)
  $z \leftarrow$ an empty solution
  $\mathcal{I}_z = \emptyset$ (set of currently covered rows)
  $\mathcal{J}_z = J$ (set of eligible columns)
  for $i \in \mathcal{I}$ **unless** $i \in \mathcal{I}_z$ **do**
    $\mathcal{C}_p \leftarrow \arg\max_{\mathcal{C}_j \in \mathcal{J}_i} |\mathcal{C}_j \setminus \mathcal{I}_z|$ (column with the most uncovered elements)
    $\mathcal{I}_z \leftarrow \mathcal{I}_z \cup \mathcal{C}_p$
    $\mathcal{J}_z \leftarrow \mathcal{J}_z \setminus \{\mathcal{C}_p\}$
  end for
  return z
end procedure

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

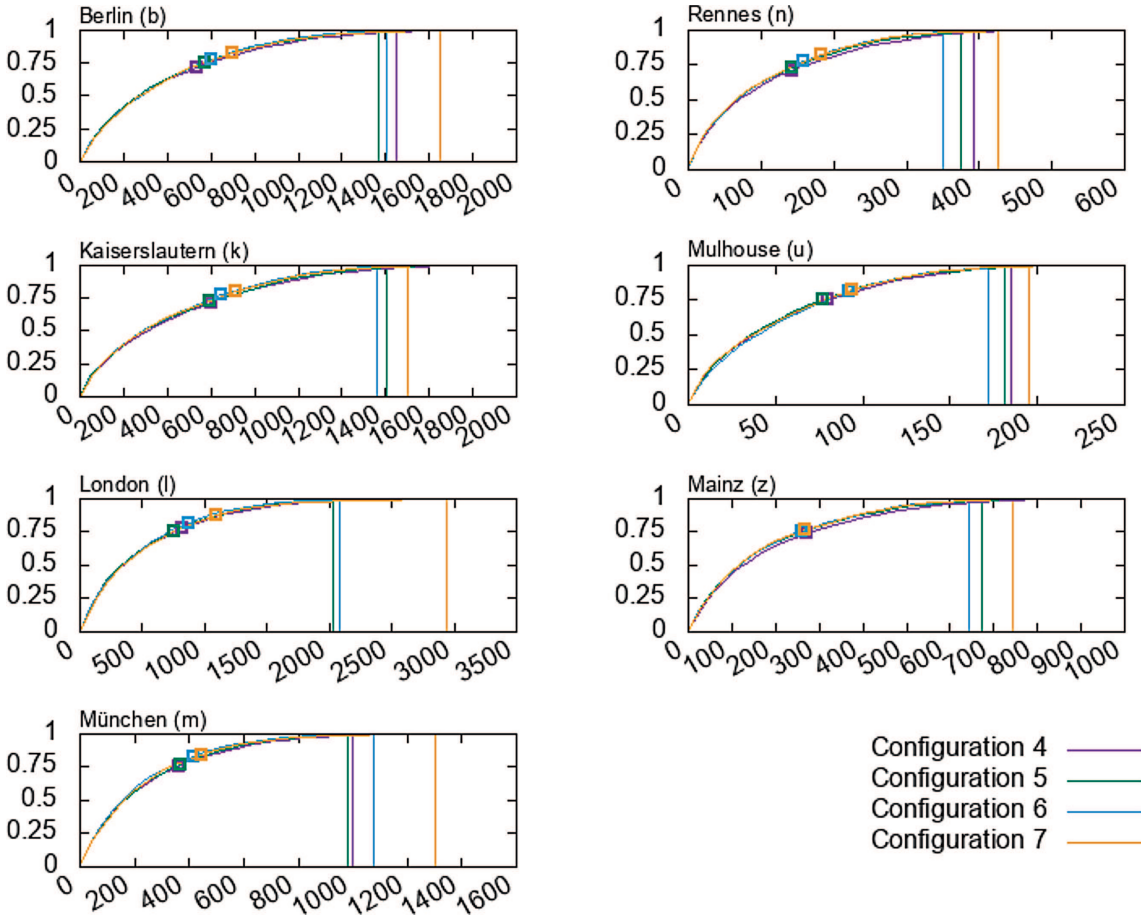

FIGURE 4. Max-k-cover solutions fronts (solution costs and coverage rates).

constraints (2.2). Two questions can therefore be raised. First, the impact of the sampling frequencies on actual area coverage is hidden by the model, which does not perceive anything outside of its samples. As a consequence, 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. In the following sections, we propose an approach to the issues above and attempt to confirm the validity of the SCP and MCP models for our problem.

6. Applicability analysis and simulations in continuous space

6.1. Method

In order to confirm the observations made in Section 5, we propose to use agent simulation algorithms in order to measure not only the impact of a tolerance on coverage but also that of the sampling parameters once the process strays away from the discrete SCP model and is evaluated in continuous, simulated space.

For each city, we therefore create a simulation scene which includes all known obstacles from the OSM data. An OCP solution also needs to be selected, which may or may not provide full coverage or optimality. Agents

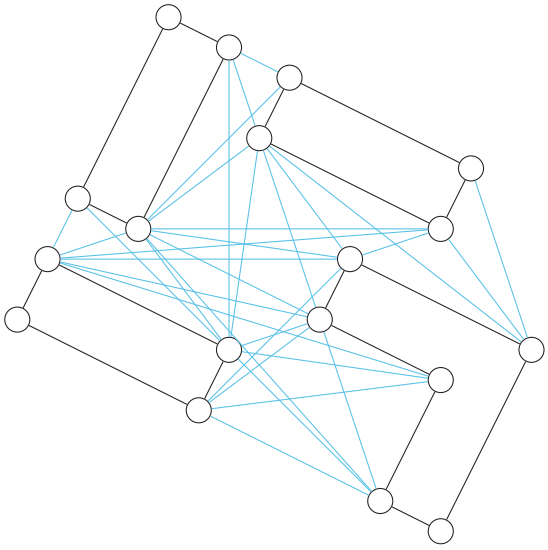

FIGURE 5. An example routing graph with wall and visibility edges.

are then created and given a start location and a destination. At every step (1 s simulation time) agents move 1.4 m, the average human walking speed, which seems appropriate given our application. Before the iteration ends, the number of visible agents is recorded. We chose to perform 3600 iterations (1 h simulation time).

In order for the agents to find their way around the city, a routing system also had to be designed. This algorithm first simplifies the geometry of the obstacles to create so-called aggregates. These correspond to connected components in the graph drawn by building walls and other obstacles. At every node, a visibility algorithm determines which other nodes are reachable in a straight line and adds the associated edges. This corresponds to a simple occlusion check which ensures no obstacles can prevent the agents from moving in the given direction. The result is a graph in which the nodes are the outer vertices of the city's buildings and the edges are either walls or the aforementioned visibility edges (see *e.g.* Fig. 5). Because this graph is typically very dense in terms of edges, agents are allowed to walk virtually anywhere in practice, under the sole constraint that they should always follow the shortest path in this graph. Therefore, when an agent is created, two endpoints are chosen randomly on building walls and a path is built by running the A^* algorithm [13, 19] on the graph. After the right number of steps, when an agent reaches its destination, it is recreated in the same fashion with a new path.

Finally, an auto-scaling algorithm was written which automatically determines a number of agents for every city. The computation is based on the approximate outdoor area of the city. The number of agents is then computed as a fraction of that area. Simulation experiments ran with graphical interfaces showed that using a density of 0.0035 agents per square meter was a reasonable albeit empirical assumption to make which maintains uniformity across our cities.

6.2. The impact of the sampling frequencies

A common and understandable assumption to make when working on the discrete optimal camera problem is that as the sampling frequencies get finer, the full-coverage solution can be expected to reach a limit corresponding to the optimal solution in a continuous environment. For this reason, the first round of simulations for this work focused on the variations of the effective coverage rate (*i.e.* the proportion of visible agents as observed in the simulation) under varying sampling configurations.

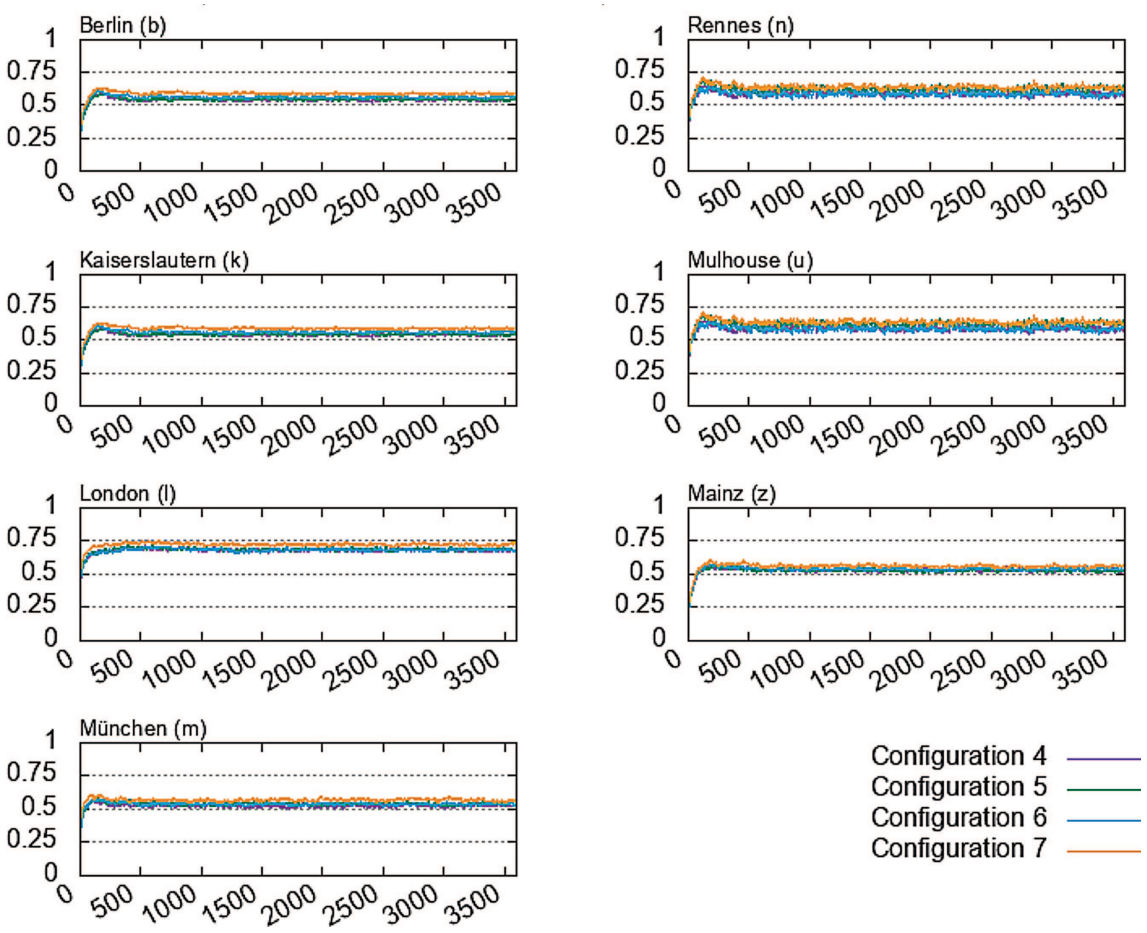

FIGURE 6. Effective coverage rates when using full coverage solutions (over 3600 simulation steps).

We propose the following approach to visualise the impact of these parameters. For every city and every configuration, the greedy full coverage solution is computed and used as input for a simulation run. At every step, the proportion of visible agents, which we call the effective coverage rate, is recorded. Figure 6 brings together the results for all our cities, which tend to be rather stable, save for the first few iterations, during which agents are exiting their buildings and entering open areas.

This figure clearly highlights the following: finer sampling frequencies do not justify their growing computational cost. Indeed, while generating and solving instances in configurations 6 and 7 takes significantly more time (see Tab. 10), the effective coverage rates when measured through simulations display very little improvement. This is mostly due to the design of our sampling algorithms, which are tailored to generate SCP samples following every city's infrastructural layout, therefore yielding a sufficiently representative sampling set.

6.3. Balancing cost and coverage

A second round of simulations was dedicated to the study of the full coverage constraint itself, and more specifically of the trade-off range which we observed on theoretical (SCP/MCP) coverage rates in Section 5.2. To this end, we propose to generate additional solutions in the 70–100% theoretical coverage rate section of the fronts (see Fig. 4) and run simulations to allow a comparison between theoretical and effective coverage

measurements. An issue arises however: while the max-k-cover greedy algorithm is able to find a solution from a given number of cameras, it cannot take a coverage rate as input. 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 3).

Algorithm 3. The binary search algorithm for fixed-coverage solutions.

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($\mathcal{I}, \mathcal{J}, k$)
    $r' \leftarrow \frac{\left| \bigcup_{\mathcal{C}_j \in z'} \mathcal{C}_j \right|}{|\mathcal{I}|}$
    if $r' < r$ **then**
      $z^- \leftarrow z'$
    else
      $z^+ \leftarrow z'$
    end if
  end while
  return z^+
end procedure

Figure 7 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 conclusion reached in Section 6.2, we now only consider our cheapest configuration: number 4.

The reported theoretical coverage rates correspond to the fronts presented earlier in Figure 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 together.

Should these gaps in the available data be filled, the simulations make it safe to assume that the OCP/MCP 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 effective MCP greedy algorithm can be used to yield appropriate solutions. This method effectively attempts to provide its users with good and applicable solutions and lifts the optimality requirement when applying the OCP to the video coverage of urban areas.

7. Conclusion and future work

In this paper, we built on top of a review of both OCP and SCP literature and designed a pipeline for real-world instance generation. The results enabled us to run a benchmark of both OCP and SCP literature

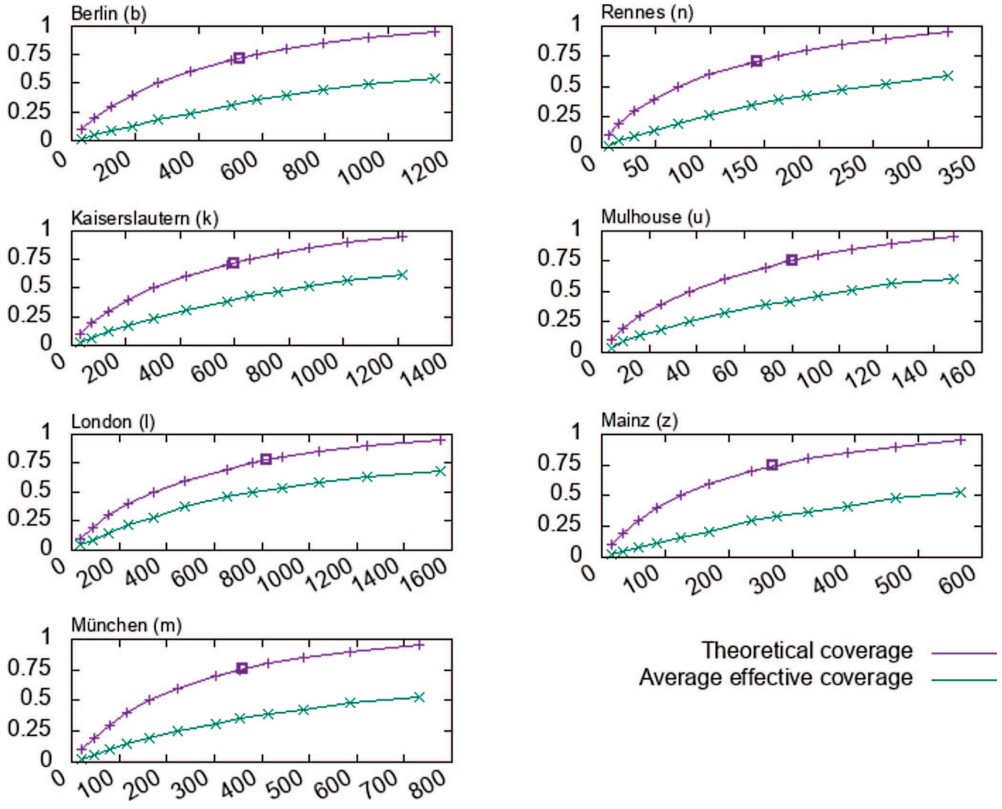

FIGURE 7. Theoretical and average effective coverage rates in configuration 4.

on the problem of global area coverage. They confirmed the state of the art but also highlighted significant computational costs for both instance generation and solving. Two key elements of the method's complexity were therefore questioned: the need for fine sampling and the impact of a rigid full coverage constraint. An approach was therefore designed to identify possible trade-offs which would enable decision makers to acquire cheaper OCP solutions in reasonable time by allowing for some tolerance on the coverage rates and by tailoring sampling procedures. Our theoretical results were then validated through simulations in continuous space, which allowed us to outline a methodology for balancing cost and coverage when designing camera-based surveillance infrastructure.

Future work on the topic will first include additional decision-making tools which not only display the range of possible trade-offs but also evaluate each solution on the fronts with regards to practical considerations. More emphasis will also have to be put on the idea of weighted subareas, so that decision makers can have more flexibility when deciding where coverage trade-offs should be made geographically speaking. Finally, improvements can still be made to the instance generation pipeline, more specifically to its visibility analysis algorithms, perhaps in collaboration with researchers in computer vision or crowd simulation, and with feedback from law enforcement officers.

Acknowledgements. This work was funded as part of the OPMoPS project by the Agence Nationale de la Recherche (France) under grant number ANR-16-SEBM-0004. Our thanks also go to the contributors and administrators of the Tier-2 high performance computing cluster at Unistra, Strasbourg, France.
Conflicts of interest. The authors have no conflict of interest to declare.

References

[1] T. Andersen and S. Tirthapura, Wireless sensor deployment for 3D coverage with constraints. In: 2009 Sixth International Conference on Networked Sensing Systems (INSS). IEEE (2009).

[2] F. Angella, L. Reithler and F. Gallesio, Optimal deployment of cameras for video surveillance systems. In: 2007 IEEE Conference on Advanced Video and Signal Based Surveillance. IEEE (2007).

[3] J. Beasley, An algorithm for set covering problem. *Eur. J. Oper. Res.* **31** (1987) 85–93.

[4] J.E. Beasley, OR-library: distributing test problems by electronic mail. *J. Oper. Res. Soc.* **41** (1990) 1069–1072.

[5] J.E. Beasley, A lagrangean heuristic for set covering problems. In: Combinatorial Optimization. Springer Berlin Heidelberg (1992) 325–326.

[6] J.E. Beasley and K. Jørnsten, Enhancing an algorithm for set covering problems. *Eur. J. Oper. Res.* **58** (1992) 293–300.

[7] R. Bodor, P. Schrater and N. Papanikolopoulos, Multi-camera positioning to optimize task observability. In: Proceedings IEEE Conference on Advanced Video and Signal Based Surveillance, 2005. IEEE (2005).

[8] M. Brévilliers, J. Lepagnot, J. Kritter and L. Idoumghar, Parallel preprocessing for the optimal camera placement problem. *Int. J. Model. Optim.* **8** (2018) 33–40.

[9] M. Brévilliers, J. Lepagnot, L. Idoumghar, M. Rebai and J. Kritter, Hybrid differential evolution algorithms for the optimal camera placement problem. *J. Syst. Inf. Technol.* **20** (2018) 446–467.

[10] V. Chvátal, A greedy heuristic for the set-covering problem. *Math. Oper. Res.* **4** (1979) 233–235.

[11] N. Conci and L. Lizzi, Camera placement using particle swarm optimization in visual surveillance applications. In: 2009 16th IEEE International Conference on Image Processing (ICIP). IEEE (2009).

[12] C.K. Cowan and P.D. Kovesi, Automatic sensor placement from vision task requirements. *IEEE Trans. Pattern Anal. Mach. Intell.* **10** (1988) 407–416.

[13] E.W. Dijkstra, A note on two problems in connexion with graphs. *Numer. Math.* **1** (1959) 269–271.

[14] U.M. Erdem and S. Sclaroff, Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Comput. Vision Image Understand.* **103** (2006) 156–169.

[15] E.P.C. Fantini and L. Chaimowicz, Coverage in Arbitrary 3D Environments: The Art Gallery Problem in Shooter Games. IEEE (2013).

[16] U. Feige, A threshold of ln n for approximating set cover. *J. ACM* **45** (1998) 634–652.

[17] T.A. Feo and M.G. Resende, A probabilistic heuristic for a computationally difficult set covering problem. *Oper. Res. Lett.* **8** (1989) 67–71.

[18] C. Gao, X. Yao, T. Weise and J. Li, An efficient local search heuristic with row weighting for the unicost set covering problem. *Eur. J. Oper. Res.* **246** (2015) 750–761.

[19] P. Hart, N. Nilsson and B. Raphael, A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **4** (1968) 100–107.

[20] E. Horster and R. Lienhart, Approximating optimal visual sensor placement: In 2006 IEEE International Conference on Multimedia and Expo. IEEE (2006).

[21] IBM®. IBM CPLEX Optimiser. https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer. Accessed: 2018-12-03.

[22] S. Indu, S. Chaudhury, N. Mittal and A. Bhattacharyya, Optimal sensor placement for surveillance of large spaces. In: 2009 Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC). IEEE (2009).

[23] R.M. Karp, Reducibility among combinatorial problems. In: Complexity of Computer Computations. Springer US (1972) 85–103.

[24] K.R. Konda and N. 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 (2013).

[25] J. Kritter, M. Brévilliers, J. Lepagnot and L. Idoumghar, On the optimal placement of cameras for surveillance and the underlying set cover problem. *Appl. Soft Comput.* **74** (2019) 133–153.

[26] J. Kritter, M. Brévilliers, J. Lepagnot and L. Idoumghar, 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 (2019).

[27] J. Kritter, M. Brévilliers, J. Lepagnot and L. Idoumghar, 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 (2020).

[28] G. Lan, G.W. DePuy and G.E. Whitehouse, An effective and simple heuristic for the set covering problem. *Eur. J. Oper. Res.* **176** (2007) 1387–1403.

[29] J. Liu, S. Sridharan and C. Fookes, Recent advances in camera planning for large area surveillance. *ACM Comput. Surv.* **49** (2016) 1–37.

[30] E. Marchiori and A. Steenbeek, An iterated heuristic algorithm for the set covering problem. In: 2nd Workshop on Algorithm Engineering (WAE98). Saarbreucken (1998) 155–166.

[31] MeshLab. http://www.meshlab.net. Accessed: 2018-12-03.

[32] T. Möller and B. Trumbore, Fast, minimum storage ray-triangle intersection. *J. Graphics Tools* **2** (1997) 21–28.

[33] Y. Morsly, M.S. Djouadi and N. Aouf, On the best interceptor placement for an optimally deployed visual sensor network. In: 2010 IEEE International Conference on Systems, Man and Cybernetics. IEEE (2010).

[34] V.P. Munishwar and N.B. Abu-Ghazaleh, Coverage algorithms for visual sensor networks. *ACM Trans. Sensor Netw.* **9** (2013) 1–36.

[35] A.T. Murray, K. Kim, J.W. Davis, R. Machiraju and R. Parent, Coverage optimization to support security monitoring. *Comput. Environ. Urban Syst.* **31** (2007) 133–147.

[36] G. Olague and R. Mohr, Optimal camera placement for accurate reconstruction. *Pattern Recogn.* **35** (2002) 927–944.

[37] OpenStreetMap, Elements – OpenStreetMap wiki. https://wiki.openstreetmap.org/wiki/Elements. Accessed: 2018-12-03.

[38] OpenStreetMap. Map features – OpenStreetMap wiki. https://wiki.openstreetmap.org/wiki/Map_Features. Accessed: 2018-12-03.

[39] J. O'Rourke, Art Gallery Theorems and Algorithms. In: Vol. 3 of *International Series of Monographs on Computer Science*. Oxford University Press (1987).

[40] OSM2World, OSM2World. http://osm2world.org. Accessed: 2018-12-03.

[41] M. Rebai, M.L. Berre, F. Hnaien and H. Snoussi, Exact biobjective optimization methods for camera coverage problem in three-dimensional areas. *IEEE Sensors J.* **16** (2016) 3323–3331.

[42] V. Satopaa, J. Albrecht, D. Irwin and B. Raghavan, Finding a "kneedle" in a haystack: detecting knee points in system behavior. In: 2011 31st International Conference on Distributed Computing Systems Workshops. IEEE (2011).

[43] M. Yagiura, M. Kishida and T. Ibaraki, A 3-flip neighborhood local search for the set covering problem. *Eur. J. Oper. Res.* **172** (2006) 472–499.

[44] Y. Yao, C.-H. Chen, B. Abidi, D. Page, A. Koschan and M. Abidi, Sensor planning for automated and persistent object tracking with multiple cameras. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition. IEEE (2008).

[45] H. Zhang, L. Xia, F. Tian, P. Wang, J. Cui, C. Tang, N. Deng and N. Ma, An optimized placement algorithm for collaborative information processing at a wireless camera network. In: 2013 IEEE International Conference on Multimedia and Expo (ICME). IEEE (2013).