Documentation of the Benchmark on the Optimal Camera Placement Problem (OCP) and the Unicost Set Covering Problem (USCP)

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

Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France

June 25, 2021

[*]Corresponding Author: mathieu.brevilliers@uha.fr

Contents

1 Introduction

1.1 OCP and USCP

The use of camera networks is now common to perform various surveillance tasks. These networks can be implemented together with intelligent systems that analyze video footage, for instance, to detect events of interest, or to identify and track objects or persons. According to [10], whatever the operational needs are, the quality of service depends on the way in which the cameras are deployed in the area to be monitored (in terms of position and orientation angles). Moreover, due to the prohibitive cost of setting or modifying such a camera network, it is required to provide a priori a configuration that minimizes the number of cameras in addition to meeting the operational needs. In this context, the optimal camera placement problem (OCP) is of critical importance, and 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 the set of positions and orientations which best (optimally) meets the requirements be determined?

More specifically, in this benchmark testbed, the objective will be to determine camera locations and orientations which ensure complete coverage of the area while minimizing the cost of the infrastructure. To this aim, a discrete approach is considered here: the surveillance area is reduced to a set of three-dimensional sample points to be covered, and camera configurations are sampled into so-called candidates each with a given set of position and orientation coordinates. A candidate can have several samples within range, and a sample can be seen by several candidates. Now, the OCP comes down to select the smallest subset of candidates which covers all the samples.

According to [7], the OCP is structurally identical to the unicost set covering problem (USCP), which is one of Karp's well-known NP-hard problems [6]. The USCP can be stated as follows: given a set of elements I (rows) to be covered, and a collection of sets J (columns) such that the union of all sets in J is I, find the smallest subset $C \subset J$ such that $\bigcup_{e \in C} e = I$. In other words, identify the smallest subset of J which covers I. As pointed out in [7], many papers dealing with the OCP use this relationship implicitly, but few works done on the USCP have been applied or adapted to the OCP, and vice versa. In very recent years however, approaches from the USCP literature have been successfully applied in the OCP context on both academic [4, 3, 9, 11, 12] and real-world [8, 9, 11, 12] problem instances. These works suggest that bridges can be built between these two bodies of literature to improve the results obtained so far on both USCP and OCP problems.

1.2 Motivations

Firstly, the main goal of this benchmark testbed is to encourage innovative research works in this direction, by proposing to solve OCP problem instances stated as USCP. Secondly, until now, no benchmark has been established for the OCP, which makes difficult to provide a fair comparison of all various propositions from the OCP literature [7]: the benchmark proposed here aims at addressing this need. Thirdly, it is a way of attracting the interest of the scientific community in new challenging USCP problem instances, that are directly related to a difficult real-world set covering problem. Actually, to the best of our knowledge, the last real-world challenging instances on the SCP were introduced in a competition called FASTER (Ferrovie Airo Set covering TendER), jointly organized by the Italian railway company (Ferrovie dello Stato SpA)and the Italian Operational Research Society (AIRO) in 1994 [5], and whose problem instances are now part of Beasley's standard OR library [2].

1.3 The benchmark testbed

This benchmark testbed gathers 69 OCP problem instances. 32 of them are academic problems similar to those tackled in [3]: various sizes and discretizations of an empty room modeled by a rectangular cuboid with cameras on the ceiling. And 37 of them are real-world problems similar to those tackled in [8]: various sizes and discretizations of urban areas with cameras on the walls of the buildings.

Actually, a subset of the academic problems listed in this benchmark was first introduced in [3], and the method used to generate all real-world instances of this benchmark is presented in [8]. For these reasons, any publication related to this benchmark testbed should refer to it by citing the articles [3, 8] and by adding a link to the benchmark website: `http://www.mage.fst.uha.fr/brevilliers/ocp-uscp-benchmark/`.

The whole benchmark testbed has been introduced for the GECCO 2020 Competition on the optimal camera placement problem (OCP) and the unicost set covering problem (USCP)[1,2]. A second edition of this competition has been hosted by GECCO 2021[3,4].

All the data files are available for download on the benchmark website, and the next two sections provide detailed information about the problem modelling, the problem instances and the instance file format for both academic and real-world problems, respectively.

[1]`https://gecco-2020.sigevo.org/index.html/Competitions`

[2]`http://www.mage.fst.uha.fr/brevilliers/gecco-2020-ocp-uscp-competition/`

[3]`https://gecco-2021.sigevo.org/Competitions`

[4]`http://www.mage.fst.uha.fr/brevilliers/gecco-2021-ocp-uscp-competition/`

2 Academic instances

A first set of 32 artificially generated instances is provided: given the technical specifications of a camera, given a three-dimensional area to monitor, and given the operational need to meet, the objective is to find a minimum set of locations (i.e. position and angular orientation) of this type of camera that ensures a total coverage of this area according to the requested operational need. The next subsection explains in detail the model used for these academic instances (which is similar to the model defined in [3]).

2.1 Problem modelling

The monitored area is defined as a rectangular box whose point coordinates (in meters) range from $(0,0,0)$ to $(X_{max}, Y_{max}, Z_{max})$ in a Cartesian coordinate system of the three-dimensional Euclidean space R^3. This area is discretized and approximated by a regular grid of sample points with a step size U (in meters) between two adjacent samples.

A camera is defined by the following technical specifications: its horizontal resolution H_{res}, its vertical resolution V_{res}, and its horizontal field of view H_{fov} (angle in degrees). It has a pyramid of vision, whose base is a rectangle with length $\frac{H_{res}}{OpNeed}$ and width $\frac{V_{res}}{OpNeed}$ (in meters), where $OpNeed$ is the operational need to be met (in pixels per meter). The height of this right pyramid corresponds to the maximal depth of view D_{max} of the camera (in meters), which depends on the operational need. Figure 1 clearly illustrates the horizontal field of view H_{fov} and the height D_{max} of the pyramid of vision. D_{max} is computed with the following equation:

$$D_{max} = \frac{\frac{1}{2} \times \frac{H_{res}}{OpNeed}}{tan\left(\frac{H_{fov}}{2} \times \frac{\pi}{180}\right)} \tag{1}$$

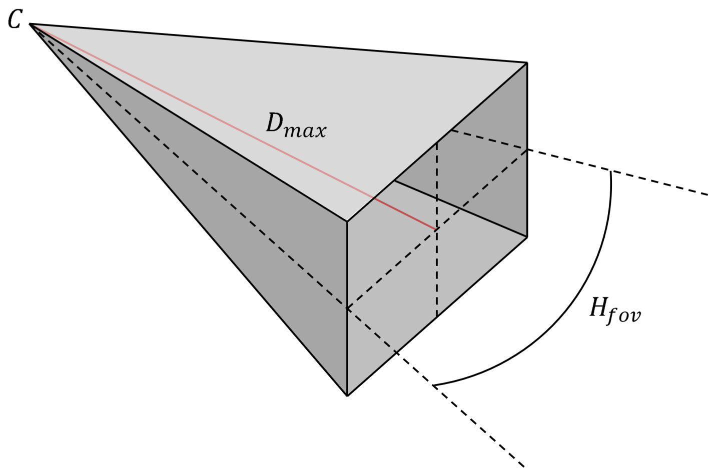

Figure 1: Example of a camera C with horizontal field of view H_{fov}, and whose pyramid of vision has height D_{max}.

A candidate (i.e. a camera location) is characterized by a point in the considered discrete grid together with discrete pan and tilt angles. Candidate coordinates (in meters) can range from $(0, 0, Z_{cam})$ to $(X_{max}, Y_{max}, Z_{cam})$ with step size U (in meters). The angular orientation of a candidate is then given by two angles: α is the pan angle, that is the rotation angle of the candidate along the Z axis, and β is the tilt angle, that is the rotation angle along the Y axis (see Figure 2). Values of α and β are discretized with the help of a parameter A, which fixes the step size to the value $\frac{\pi}{A}$. It means that α can take $N_\alpha = 2A$ different values that range in $[0, 2\pi[$. Regarding β, we only consider $N_\beta = \lfloor A/2 \rfloor + 1$ different values that range in $[0, \lfloor A/2 \rfloor \times \frac{\pi}{A}]$, given that candidates are above the samples (and thus have to be oriented downward), and given that any candidate with pan angle α and tilt angle $\beta = k \times \frac{\pi}{A}$ such that $\beta < \frac{\pi}{2}$, will be identical to the candidate with same coordinates and pan angle $\alpha' = \alpha + \pi$ and tilt angle $\beta' = \pi - k \times \frac{\pi}{A}$.

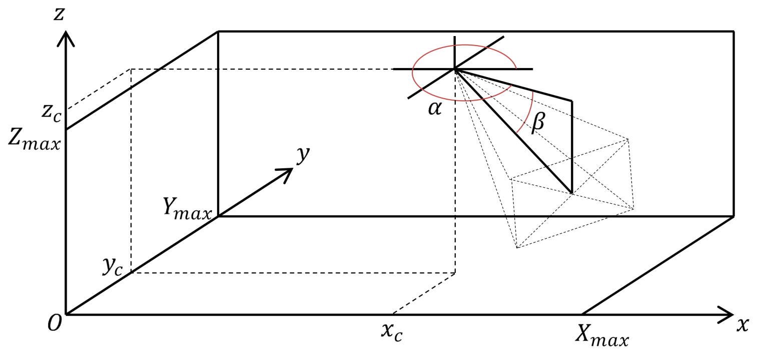

Figure 2: Example of candidate with coordinates (x_c, y_c, z_c), pan angle α, and tilt angle β.

If the samples are labelled with integers (representing the set of elements to be covered), then any candidate can be modelled as a set of integers (corresponding to the labels of the samples it covers), and the OCP can be formulated as a USCP in a straightforward manner: given the set I of elements (i.e. samples) and a collection J of sets (i.e. candidates), solving the OCP comes down to find the minimum subset of J that covers I.

Now, the following decision variables can be defined:

$$\forall c \in J,\ x_c = \begin{cases} 1 & \text{if candidate } c \text{ is used,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Then, the corresponding binary integer linear programming model can be written as follows:

$$Min \sum_{c \in J} x_c \tag{3}$$

subject to

$$\forall s \in I,\ \sum_{c \in J : s \in c} x_c \geq 1 \tag{4}$$

$$\forall c \in J,\ x_c \in \{0, 1\}\,. \tag{5}$$

The objective function (see Equation 3) minimizes the total number of used candidates. Equation 4 indicates that each sample has to be covered by at least one candidate (full coverage constraint). Equation 5 gives the set of binary constraints needed for the decision variables defined in Equation 2.

2.2 Problem instances

Table 1 shows the specifications of the 32 academic instances: instance name, X_{max}, Y_{max}, Z_{max}, Z_{cam}, $OpNeed$, H_{res}, V_{res}, H_{fov}, U, A, number of samples (rows), number of candidates (columns). **It is worth noting that these instances can be significantly reduced before optimization by using classical procedures, i.e. domination and inclusion checks [1].**

Table 1: Specifications of the 32 academic instances.

Name	X_{max}	Y_{max}	Z_{max}	Z_{cam}	$OpNeed$	H_{res}	V_{res}	H_{fov}	U	A	Samples	Candidates
AC_01	5	5	2	2.5	100	1920	1080	65	0.5	4	605	2904
AC_02	10	10	2	2.5	100	1920	1080	65	0.5	4	2205	10584
AC_03	15	15	2	2.5	100	1920	1080	65	0.5	4	4805	23064
AC_04	20	20	2	2.5	100	1920	1080	65	0.5	4	8405	40344
AC_05	25	25	2	2.5	100	1920	1080	65	0.5	4	13005	62424
AC_06	30	30	2	2.5	100	1920	1080	65	0.5	4	18605	89304
AC_07	40	40	2	2.5	100	1920	1080	65	0.5	4	32805	157464
AC_08	50	50	2	2.5	100	1920	1080	65	0.5	4	51005	244824
AC_09	60	60	2	2.5	100	1920	1080	65	0.5	4	73205	351384
AC_10	5	5	2	2.5	500	1920	1080	65	0.5	4	605	2904
AC_11	10	10	2	2.5	500	1920	1080	65	0.5	4	2205	10584
AC_12	15	15	2	2.5	500	1920	1080	65	0.5	4	4805	23064
AC_13	20	20	2	2.5	500	1920	1080	65	0.5	4	8405	40344
AC_14	25	25	2	2.5	500	1920	1080	65	0.5	4	13005	62424
AC_15	30	30	2	2.5	500	1920	1080	65	0.5	4	18605	89304
AC_16	40	40	2	2.5	500	1920	1080	65	0.5	4	32805	157464
AC_17	50	50	2	2.5	500	1920	1080	65	0.5	4	51005	244824
AC_18	60	60	2	2.5	500	1920	1080	65	0.5	4	73205	351384
AC_19	70	70	2	2.5	500	1920	1080	65	0.5	4	99405	477144
AC_20	80	80	2	2.5	500	1920	1080	65	0.5	4	129605	622104
AC_21	90	90	2	2.5	500	1920	1080	65	0.5	4	163805	786264
AC_22	100	100	2	2.5	500	1920	1080	65	0.5	4	202005	969624
AC_23	110	110	2	2.5	500	1920	1080	65	0.5	4	244205	1172184
AC_24	120	120	2	2.5	500	1920	1080	65	0.5	4	290405	1393944
AC_25	130	130	2	2.5	500	1920	1080	65	0.5	4	340605	1634904
AC_26	140	140	2	2.5	500	1920	1080	65	0.5	4	394805	1895064
AC_27	150	150	2	2.5	500	1920	1080	65	0.5	4	453005	2174424
AC_28	160	160	2	2.5	500	1920	1080	65	0.5	4	515205	2472984
AC_29	170	170	2	2.5	500	1920	1080	65	0.5	4	581405	2790744
AC_30	180	180	2	2.5	500	1920	1080	65	0.5	4	651605	3127704
AC_31	190	190	2	2.5	500	1920	1080	65	0.5	4	725805	3483864
AC_32	200	200	2	2.5	500	1920	1080	65	0.5	4	804005	3859224

2.3 Instance file format

AC_XX_cover.txt This file contains all the needed information to solve the problem instance AC_XX as a strict USCP:

- number of samples and number of candidates,
- then, for each sample s: its integer label s, the number of candidates which cover sample s, and a list of the candidates which cover sample s.

In addition to these cover information, geometric information regarding the problem instances are provided for those who aim to use an OCP-specific algorithm.

AC_specs.txt This file contains the geometric specifications of each instance, one instance per line, in the following order: instance name, $OpNeed$, H_{res}, V_{res}, H_{fov}, U, and A.

AC_XX_samples.txt This file contains the discrete coordinates (in the regular grid, see Section 2.1) of each sample point of problem instance AC_XX, with the following format:

- number of samples,
- then, for each sample s: its integer label, and the integer coordinates (x_s, y_s, z_s) of s in the regular grid which approximate the monitored area.

For convenience, this file stores the coordinates of each sample s in the regular grid, but the real coordinates of s can be easily retrieved with the help of the step size U:

$$(x_s \times U, y_s \times U, z_s \times U). \tag{6}$$

AC_XX_candidates.txt This file contains the discrete coordinates (see Section 2.1) of each candidate camera of problem instance AC_XX, with the following format:

- number of candidates,
- then, for each candidate c: its integer label, and the integer coordinates $(x_c, y_c, z_c, \beta_c, \alpha_c)$ of c.

For convenience, this file stores the discrete coordinates of each candidate c, but the real coordinates of c can be easily retrieved with the help of the step size U and the parameter A:

$$(x_c \times U, y_c \times U, z_c \times U, \beta_c \times \frac{\pi}{A}, \alpha_c \times \frac{\pi}{A}). \tag{7}$$

3 Real-world instances

A second set of 37 real-world instances can also be found alongside the academic ones. The main difference lies in the fact that these instances have been generated using map and elevation data from actual urban areas. The objective remains the same: to find the minimum set of camera configurations (position and orientation) which ensures full coverage. The following sections introduce these instances and the method used to generate and model them.

3.1 Problem modelling

For these instances, the area to be covered is no longer described by a regular shape. The points to be covered are still represented in a 3D Cartesian coordinate system using the meter as the unit. A point is therefore a simple (x, y, z) triple. The sampling procedure is based on map and elevation data and follows the area's local infrastructure, meaning no regular position pattern should be expected as far as the points are concerned.

Cameras are modelled using the same method as for the academic instances: their coverage is defined by their resolution, their field of view and an operational need parameter set in pixels-per-meter. The range and frustum computations are roughly identical. The final representation of a configuration is however different, as it is impossible to work on a regular grid when placing cameras on existing city infrastructure. For this reason, in these instances, a camera configuration is defined by two triples: one for position (x_p, y_p, z_p) and one for orientation (x_o, y_o, z_o). The former follows the same semantics as for points, while the latter is a unit vector oriented to point towards the centre of the frustum pyramid's base when attached at the camera's position (see the red line in Figure 1).

While samples are not generated uniformly, one will most likely notice some patterns in both positions and orientations which can be attributed to the inner-workings of the sampling procedure. The latter uses maps to determine where points should be placed and does so at regular intervals along elements such as roads, alleyways, parking lots, open areas and so on. This procedure requires various parameters, most of which are sampling frequencies which answer questions such as "how often along a road (polyline) should a point be created?" The same applies to the sampling of camera configurations, which follows existing buildings, walls, poles and other such elements. Orientation angles are sampled at regular intervals of $\frac{\pi}{5}$ or $\frac{\pi}{6}$ for panning and $\frac{\pi}{10}$ or $\frac{\pi}{12}$ for tilting. The bounds are similar to those of the academic instances. Points and camera positions were generated every 3 or 5 meters (depending on the instance) along the lines of the city's geometry. For more detailed information about the sampling procedure, the reader is referred to [8].

Aside from the sampling procedure and the resulting model, the real-world instances should be solved for the same objective and under the same constraints as the academic ones. The unicost set covering problem model given by Equations (3), (4) and (5) is therefore applicable here and the visibility matrices follow the exact same format in the instance files.

3.2 Problem instances

Table 2 gives basic statistics on the 37 real-world instances. **The instances have been reduced by applying work reported in [1].** The description is identical to that of the academic instances, save for the grid parameters which do not apply here.

Table 2: Specifications of the 37 real-world instances.

Name	$OpNeed$	H_{res}	V_{res}	H_{fov}	Samples	Candidates
RW_01	25	1920	1080	65	153368	32430
RW_02	25	1920	1080	65	285698	56132
RW_03	25	1920	1080	65	161099	32040
RW_04	25	1920	1080	65	304655	59137
RW_05	25	1920	1080	65	206900	34568
RW_06	25	1920	1080	65	380420	65691
RW_07	25	1920	1080	65	214889	42046
RW_08	25	1920	1080	65	382651	77986
RW_09	25	1920	1080	65	206816	39003
RW_10	25	1920	1080	65	368114	71323
RW_11	25	1920	1080	65	82437	15632
RW_12	25	1920	1080	65	136555	28109
RW_13	25	1920	1080	65	293138	61741
RW_14	25	1920	1080	65	81062	14916
RW_15	25	1920	1080	65	141309	27008
RW_16	25	1920	1080	65	105829	21063
RW_17	25	1920	1080	65	180453	35635
RW_18	25	1920	1080	65	79947	14423
RW_19	25	1920	1080	65	141114	26483
RW_20	25	1920	1080	65	332300	50284
RW_21	25	1920	1080	65	654068	90050
RW_22	25	1920	1080	65	83835	17203
RW_23	25	1920	1080	65	142326	31038
RW_24	25	1920	1080	65	201967	33880
RW_25	25	1920	1080	65	375680	59851
RW_26	25	1920	1080	65	105566	18043
RW_27	25	1920	1080	65	181090	32669
RW_28	25	1920	1080	65	136755	27838
RW_29	25	1920	1080	65	273964	49267
RW_30	25	1920	1080	65	263518	49354
RW_31	25	1920	1080	65	472660	87248
RW_32	25	1920	1080	65	124289	30189
RW_33	25	1920	1080	65	229231	55000
RW_34	25	1920	1080	65	134479	27329
RW_35	25	1920	1080	65	238546	47590
RW_36	25	1920	1080	65	135043	28162
RW_37	25	1920	1080	65	238492	50702

3.3 Instance file format

RW_XX_cover.txt This file follows the same format as academic instances (see Section 2.3).

RW_specs.txt This file contains the operational parameters of each instance, that is, the first five columns in Table 2.

RW_XX_samples.txt This file contains the real coordinates of each sample point of problem instance RW_XX, with the following format:

- number of samples,
- then, for each sample: its integer label and its real coordinates (x, y, z) in a 3D Cartesian coordinate system (see Section 3.1).

RW_XX_candidates.txt This file contains the real coordinates of each candidate camera configuration of problem instance RW_XX, with the following format:

- number of candidates,
- then, for each candidate configuration: its integer label, its position vector (x_p, y_p, z_p) and its orientation unit vector (x_o, y_o, z_o) (see Section 3.1).

References

[1] J. E. Beasley. An algorithm for set covering problem. European Journal of Operational Research, 31(1):85 – 93, 1987. doi:https://doi.org/10.1016/0377-2217(87)90141-X.

[2] J. E. Beasley. Or-library: Distributing test problems by electronic mail. Journal of the Operational Research Society, 41(11):1069–1072, 1990. doi:10.2307/2582903.

[3] M. Brévilliers, J. Lepagnot, L. Idoumghar, M. Rebai, and J. Kritter. Hybrid differential evolution algorithms for the optimal camera placement problem. Journal of Systems and Information Technology, 20(4):446 – 467, 2018. doi:https://doi.org/10.1108/JSIT-09-2017-0081.

[4] M. Brévilliers, J. Lepagnot, J. Kritter, and L. Idoumghar. Parallel preprocessing for the optimal camera placement problem. International Journal of Modeling and Optimization, 8(1):33 – 40, 2018. doi:10.7763/IJMO.2018.V8.621.

[5] A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. Operations Research, 47(5):730–743, 1999. doi:https://doi.org/10.1287/opre.47.5.730.

[6] R. M. Karp. Reducibility among Combinatorial Problems, pages 85–103. Springer US, Boston, MA, 1972. doi:10.1007/978-1-4684-2001-2_9.

[7] J. Kritter, M. Brévilliers, J. Lepagnot, and L. Idoumghar. On the optimal placement of cameras for surveillance and the underlying set cover problem. Applied Soft Computing, 74:133 – 153, 2019. doi:https://doi.org/10.1016/j.asoc.2018.10.025.

[8] 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 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), pages 1103–1108, April 2019. doi:10.1109/CoDIT.2019.8820295.

[9] W. Lin, F. Ma, Z. Su, Q. Zhang, C. Li, and Z. Lü. Weighting-based parallel local search for optimal camera placement and unicost set covering. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, GECCO '20, pages 3–4, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3377929.3398184.

[10] J. Liu, S. Sridharan, and C. Fookes. Recent advances in camera planning for large area surveillance: A comprehensive review. ACM Comput. Surv., 49(1):6:1–6:37, May 2016. doi:10.1145/2906148.

[11] Z. Su, Q. Zhang, Z. Lü, C. Li, W. Lin, and F. Ma. Weighting-based variable neighborhood search for optimal camera placement. Proceedings of the AAAI Conference on Artificial Intelligence, 35(14):12400–12408, May 2021. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17471.

[12] Y. Wang, S. Pan, S. Al-Shihabi, J. Zhou, N. Yang, and M. Yin. An improved configuration checking-based algorithm for the unicost set covering problem. European Journal of Operational Research, 294(2):476–491, 2021. `doi:https://doi.org/10.1016/j.ejor.2021.02.015`.